A Luca
A Sandra
A Gilberto

Indice Generale

SOMMARIO

Le reti acustiche sottomarine (Underwater Acoustic Networks – UAN) rappresentano un nuovo scenario di interesse per le telecomunicazioni.

L'obiettivo di questa tesi è di presentare un nuovo approccio al problema del routing per le reti sottomarine tolleranti al ritardo (Delay Tolerant Networks – DTN). Il lavoro si colloca al livello tre dello stack ISO/OSI, quello cui i ricercatori in ambito underwater finora hanno rivolto minore attenzione, a fronte di un maggior impegno a livello fisico e datalink.

Sviluppare un nuovo protocollo di routing per UAN richiede molta attenzione, i fattori in gioco sono molteplici ed è pressoché impossibile realizzare un algoritmo efficiente in ogni situazione. La progettazione va quindi fatta ad hoc per un particolare scenario in modo da trarne i maggiori benefici possibili, tenendo ad esempio in considerazione la collocazione dei nodi, il fatto che siano mobili o meno, l'eventuale presenza di sink o AUV (Autonomous Underwater Vehicle) ecc. tutti elementi cui vanno sommati i vincoli imposti dalla trasmissione acustica.

L'algoritmo è sviluppato per reti composte da soli nodi mobili ignari della posizione altrui, si adatta ad ogni frangente, ma è vincolato dalla conoscenza a priori di una stima della statistica degli incontri tra essi.

La presentazione degli argomenti è così organizzata: nel primo capitolo verrà fornita un'introduzione generica sulle reti sottomarine, cui seguirà nel secondo capitolo un approfondimento sulle reti DTN e i relativi protocolli di routing; nel terzo capitolo è descritto il protocollo proposto in questa tesi, nel quarto verranno descritti lo scenario delle simulazioni e i relativi risultati, infine nel quinto capitolo saranno tratte alcune conclusioni e discussi eventuali sviluppi futuri.

CAPITOLO 1

LE RETI ACUSTICHE SOTTOMARINE

Le reti acustiche sottomarine sono oggetto di un crescente impiego in molti ambiti e con diverse finalità: raccolta di dati oceanografici (conformazione di coste, fondali ecc.), monitoraggio dell'inquinamento delle acque, esplorazioni in mare aperto (individuazione di giacimenti vari, di siti per la posa di cavi o strutture ecc.), prevenzione di disastri (rilevamento degli effetti delle attività umane sugli ecosistemi marini, e delle attività sismiche subacquee), navigazione assistita (rilevamento scogli od ostacoli generici a basse profondità), e sorveglianza (maggiore precisione rispetto ai tradizionali radar).

L'approccio tradizionale per adempiere a questi compiti, consiste nel dispiegare sensori che registrano dati durante la missione, per poi essere recuperati; tale approccio però non rende possibili il monitoraggio in tempo reale e l'interazione degli strumenti con i sistemi di controllo a terra; inoltre se vi sono errori, non è possibile rilevarli prima del recupero dei sensori, e la quantità di dati registrabili è limitata dalla capacità dei dispositivi di memorizzazione di bordo.

Le reti acustiche sottomarine consentono di far fronte ad alcuni dei sopra citati limiti del vecchio approccio: la comunicazione tra dispositivi permette ad esempio, il monitoraggio in tempo reale di determinate aree, la configurazione remota e l'interazione con operatori umani.

La tecnologia di base per lo scambio di informazioni tra dispositivi subacquei e per la creazione di reti acustiche sottomarine è la comunicazione wireless acustica; sott'acqua infatti, la trasmissione a lunghe distanze mediante onde radio sarebbe possibile solo a frequenze molto basse (30-300Hz), che richiedono antenne di grandi dimensioni ed elevate potenze trasmissive. Le trasmissioni a frequenze ottiche, inoltre, pur non risentendo molto di attenuazioni, sono soggette a dispersione, per cui questo tipo di segnali sott'acqua offrirebbe copertura solo a corto raggio (≤100m).

La realizzazione di una UAN funzionale non è per niente un compito facile: ai problemi di progettazione tipici delle reti normali, bisogna aggiungere diverse considerazioni dal punto di vista energetico (i dispositivi sono alimentati a batteria e le operazioni di trasmissione acustica sono molto dispendiose in termini di energia), dal punto di vista dell'affidabilità (l'ambiente sottomarino aumenta la predisposizione a guasti) e dal punto di vista del canale trasmissivo.

1.1 IL CANALE ACUSTICO SOTTOMARINO

La trasmissione acustica mediante il canale sottomarino è soggetta a:

- ritardi di propagazione maggiori di diversi ordini di grandezza rispetto al canale radio terrestre (la velocità di propagazione del suono è sensibilmente inferiore a quella delle onde elettromagnetiche in aria);
- propagazione multipath a causa dei fenomeni di rifrazione, riflessione e dispersione;
- larghezza di banda limitata tra qualche centinaio di Hz e qualche decina di kHz, a causa dei fenomeni di assorbimento e del rumore;
- significative distorsioni da effetti Doppler sui segnali ricevuti e conseguenti elevati tassi di errore sul bit;
- perdita di connettività temporanea dovuta alla variazione delle condizioni di canale.

Le prestazioni dei sistemi di comunicazione subacquei sono quindi limitate, e non è possibile aumentare arbitrariamente il bit rate o abbassare il tasso di errore sul bit, aumentando ad esempio la potenza di trasmissione.
Nel seguito verranno approfonditi gli aspetti principali da tenere in considerazione quando si parla di canale acustico sottomarino: l'attenuazione, il rumore e l'SNR.

1.1.1 Attenuazione

La componente di attenuazione, principale responsabile delle perdite di comunicazione, dipende dalla frequenza del segnale e dalla distanza tra sorgente e destinazione: i collegamenti più corti (dove è possibile utilizzare frequenze più alte) offrono più banda rispetto ai collegamenti su distanze maggiori; per esempio una trasmissione di 100 km può essere eseguita direttamente da sorgente a destinazione usando una banda da 10 a 100Hz, oppure trasmettendo l'informazione su 10 hop di 10 km ciascuno, ma con una larghezza di banda nell'ordine di qualche kHz e conseguente aumento del throughput.

Se viene stabilito un collegamento acustico sottomarino tra due nodi a distanza l , che trasmettono alla frequenza f, l'attenuazione è data dalla seguente formula:

$$A(l, f) = l^k a(f)^l \ ,$$

dove k è chiamato esponente di path-loss ed $a(f)$ coefficiente di assorbimento;

$a(f)$ può essere espresso in decibel usando la formula di Thorp che lo restituisce in dB/km dato f in kHz:

$$10\log a(f) \ = \ 0.11\frac{f^2}{1+\ f^2} + 44\frac{f^2}{4100\ +\ f^2} + 2.75 * 10^{-4} f^2 + 0.003$$

La formula precedente è valida per frequenze oltre qualche centinaio di Hz; per frequenze più basse si usa invece la seguente:

$$10\log a(f) = 0.002 + 0.11\frac{f^2}{1+f^2} + 0.011 f^2$$

Il grafico di figura 1 mostra l'andamento di $a(f)$ al variare della frequenza:

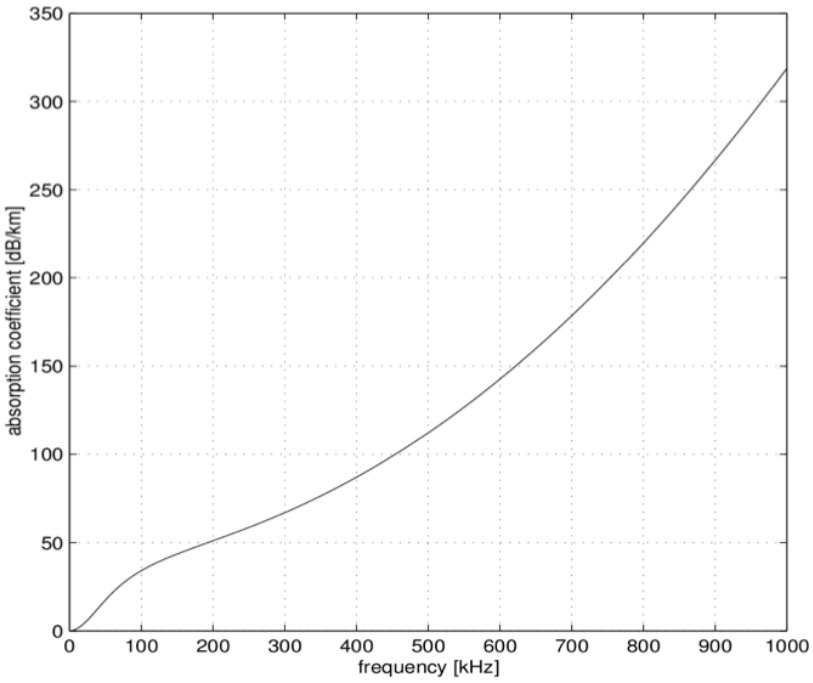


Fig. 1: Coefficiente d'assorbimento *a(f)* in [dB/km]

1.1.2 Rumore

Il rumore è generato dalla somma delle seguenti componenti: turbolenza, navigazione, vento e onde superficiali, e rumore termico, di cui possiamo modellare la densità spettrale di potenza (p.s.d.) in dB re μPa/Hz in funzione della frequenza f in kHz.

$$10 \log N_t(f) = 17 - 30 \log f$$

$$10 \log N_s(f) = 40 + 20 (s - 0.5) + 26 \log f - 60 \log (f + 0.03)$$

$$10 \log N_w(f) = 50 + 7.5\sqrt{w} + 20 \log f - 40 \log (f + 0.4)$$

$$10 \log N_{th}(f) = -15 + 20 \log f$$

In particolare solo le frequenze molto basse (<10Hz) subiscono l'effetto del rumore generato dalla componente di *turbolenza*, la componente di *navigazione* influenza invece le frequenze nell'intervallo 10Hz-100Hz e viene espressa con un fattore compreso tra 0 e 1, la componente di *rumore termico* è presente solo per frequenze > 10kHz, infine il contributo maggiore al rumore viene dato dalla componente del *vento* e delle *onde superficiali* presente sulle frequenze tra 100Hz e 100kHz, ovvero l'intervallo su cui opera la trasmissione acustica.
Nel grafico di figura 2 è illustrata la densità spettrale di potenza delle varie componenti del rumore per s = 0.5 e w = 0:

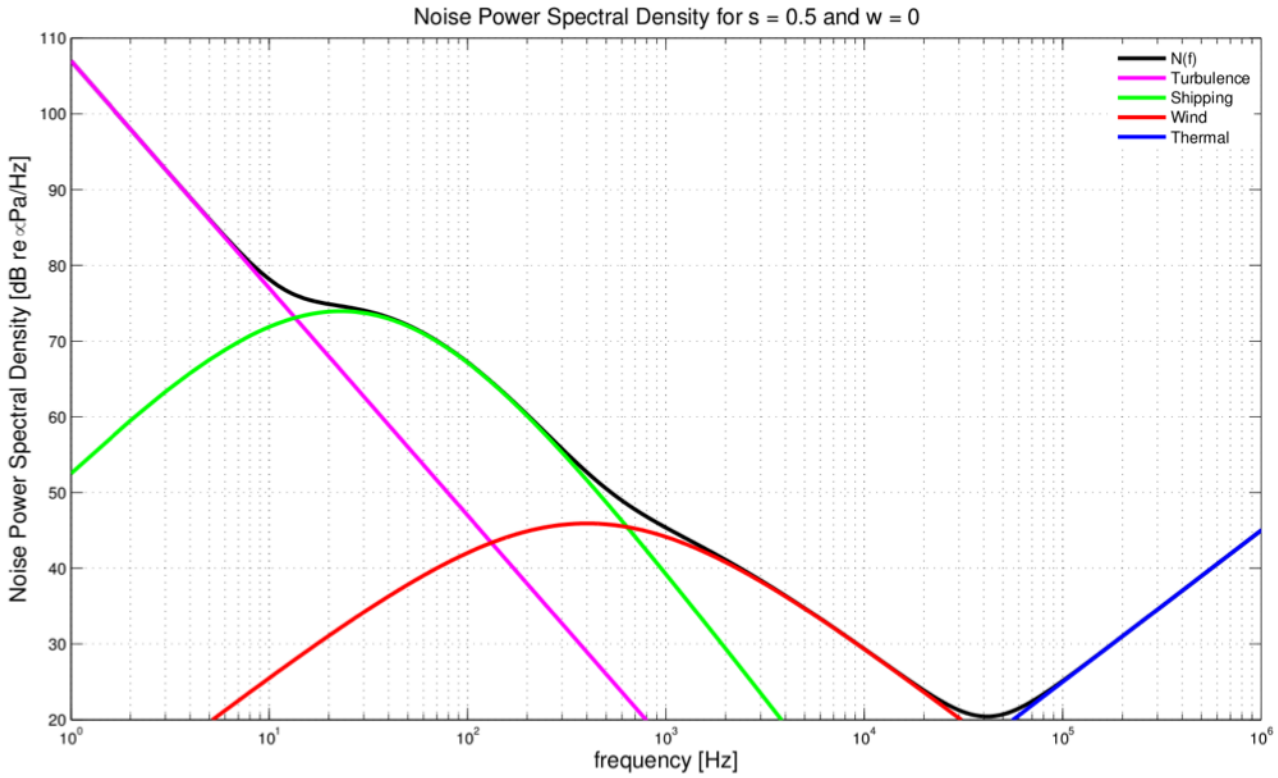


Fig. 2: densità spettrale di potenza del rumore nel canale sottomarino per s=0.5 e w=0

1.1.3 SNR

Il rapporto segnale rumore (SNR) è espresso invece dalla seguente formula:

$$SNR_{dB} = 10\, log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}}\right)$$

Usando l'attenuazione *A(l,f)* e la densità spettrale di potenza del rumore *N(f)* possiamo calcolare l'SNR tra due nodi a distanza *l* quando il segnale è trasmesso alla frequenza *f* con potenza *P*:

$$SNR\ (l,f) = \frac{P/A(l,f)}{N(f)\Delta f}$$

Dove

- $\triangle f$ è la larghezza di banda attorno alla frequenza di trasmissione f
- $\frac{1}{A(l,f)\,N(f)}$ è la componente dipendente dalla frequenza

Il Grafico di figura 3 mostra l'andamento della quantità $\frac{1}{A(l,f)\,N(f)}$ per alcune distanze, al variare della frequenza in kHz:

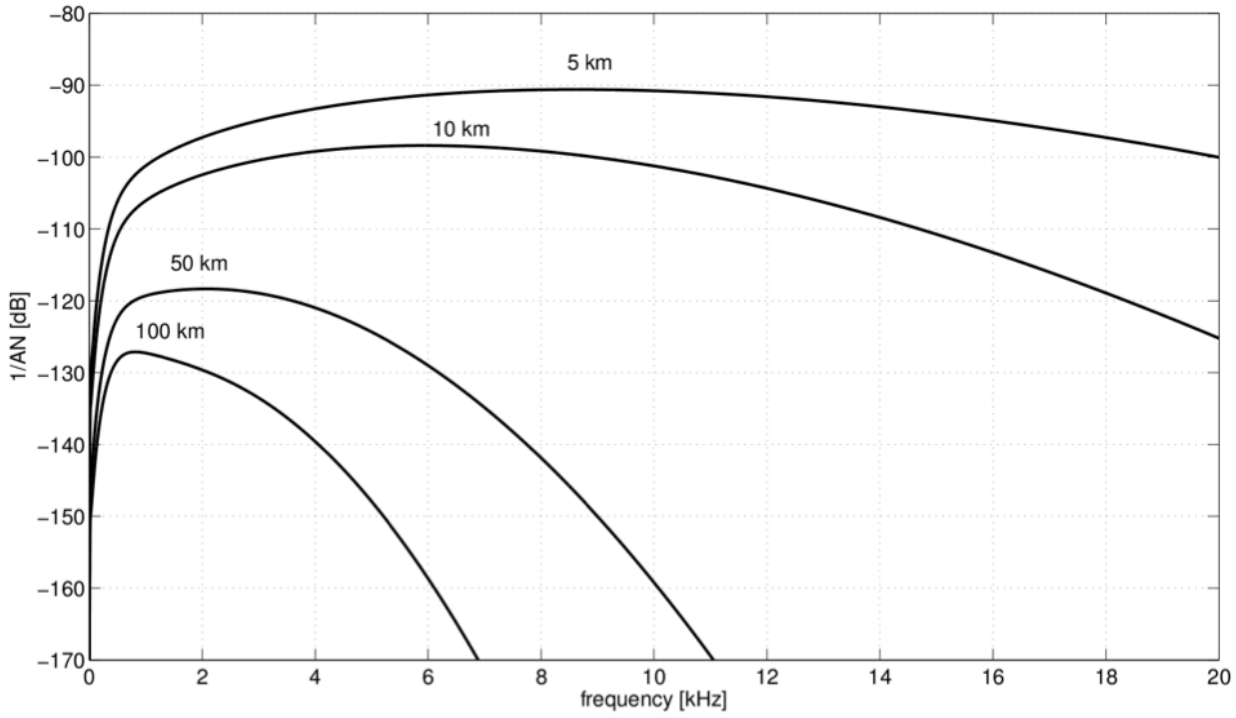


Fig. 3: La quantità *1/A(l,f)N(f)* per alcune distanze

CAPITOLO 2

PROTOCOLLI DI ROUTING

In una rete generica il routing è il processo di selezione dei percorsi multi-salto lungo cui inviare il traffico, ovvero il processo di scelta dei nodi intermedi tra sorgente e destinazione cui inoltrare i pacchetti.
Di norma l'inoltro viene gestito mediante tabelle che mantengono informazioni sui percorsi da seguire a seconda della destinazione, e la cui costruzione è di primaria importanza per un instradamento efficiente.
Il routing viene effettuato al livello di rete dello stack ISO/OSI.
La maggior parte degli algoritmi di instradamento utilizza un singolo percorso, alcuni prevedono l'utilizzo di percorsi alternativi multipli, ma molti altri sono gli aspetti che differenziano i vari approcci al routing :

- *Source routing*: il percorso è predeterminato dalla sorgente, che inserisce tali informazioni nel pacchetto. Ogni dispositivo raggiunto legge così le informazioni contenute nel pacchetto e agisce di conseguenza senza prendere alcuna decisione autonoma riguardo all'inoltro.

- *Hop-by-hop routing*: i pacchetti contengono solo l'indirizzo di destinazione e sono i dispositivi a determinare la rotta migliore per instradarli.

- *Static routing*: non c'è scambio di informazioni sulla topologia della rete tra i router, nei quali i percorsi fissati a priori sono inseriti manualmente dall'amministratore. Questo tipo di configurazione non è fault tolerant: in caso di cambiamento o guasto nella rete, non vi è alcuna compensazione o deviazione, e il traffico può riprendere a viaggiare correttamente solo dopo un'eventuale riparazione o in seguito al cambiamento delle route.

- *Reactive routing*: viene cercato un percorso solo in presenza di dati da trasmettere, con conseguente overhead per il processo di route discovery, e basso controllo del traffico.

Funziona bene in reti mobili indipendentemente dalla velocità di movimento, ma soffre in termini di ritardo venendo i pacchetti scartati finché non è stata scoperta la route. Può risultare più robusto rispetto ad altri approcci nel caso il processo di scoperta porti ad individuare più route, che verrebbero utilizzate in caso di fallimento delle precedenti. Protocolli reattivi sono ad esempio l'AODV e il DSR.

- *Proactive routing*: i percorsi tra ogni coppia di nodi vengono trovati in anticipo, e sono aggiornati periodicamente tramite scambio di informazioni sulla topologia della rete. Può soffrire in condizioni di traffico e mobilità elevate, ma offre buone prestazioni in termini di ritardo essendo i percorsi stabiliti a priori. Tra i protocolli proattivi troviamo l'OLSR e il DSDV.

- *Adaptive routing*: i sistemi sono in grado di modificare i percorsi in risposta al cambiamento di determinate condizioni (tipicamente caduta di un nodo o di un collegamento), mantenendo attive il maggior numero di route possibili. I protocolli di questo tipo più famosi sono il RIP e l'OSPF.

- *Geographic routing*: è basato sul progressivo avanzamento del pacchetto verso la destinazione, cioè la distanza del nodo che contiene il pacchetto dovrebbe diminuire dopo ogni trasmissione di successo in un percorso multihop. In una rete sufficientemente densa ogni nodo ha almeno un vicino ammissibile per trasmettere un pacchetto in base ai criteri di avanzamento geografico, in reti sparse soggette a disconnessioni invece, può essere necessario memorizzare alcune informazioni topologiche per uscire da eventuali "buchi di connettività". Altri problemi per questo algoritmo sorgono nel caso non sia nota la posizione della destinazione (in reti completamente mobili ad esempio), rendendo necessari meccanismi per informare i nodi sulle posizioni altrui che, pur se esistenti e collaudati (es: ALBA-R), vanno quasi ad azzerare in termini di overhead i vantaggi del routing geografico.

- *Hierarchical routing*: stabilisce un rapporto di subordinazione tra alcuni nodi e un insieme di nodi master. Può lavorare in due modi: organizzando i nodi in cluster o mediante l'istituzione di un albero che copre l'intera rete in base a determinati criteri. Nel primo caso qualche algoritmo determina se un nodo deve essere master (a capo del cluster) o figlio, dopo di che

tutti i nodi figli trasmettono al nodo master più vicino, il quale dovrà comunicare nel modo più efficiente possibile (ricorrendo all'organizzazione ad albero ad esempio) con gli altri nodi master. Infine, un algoritmo periodicamente ruota i nodi a capo dei cluster, in quanto particolarmente stressati dai compiti di raccolta dati e organizzazione della rete.

- *Flat routing*: i nodi assumono ruoli equivalenti nel processo di spedizione dei dati: tutti gli algoritmi geografici sono flat, ad eccezione di quelli che includono algoritmi contro i "buchi" di connettività.

- *Distance Vector routing*: basato sull'assegnazione di un costo ad ogni collegamento tra i nodi della rete, e sull'invio delle informazioni tramite il percorso dal minor costo totale. All'avvio ogni nodo conosce solo i propri vicini e i costi per raggiungerli. Scambiando questi dati con i propri vicini, tramite l'algoritmo di Bellman-Ford, il nodo costruisce una tabella che ad ogni destinazione conosciuta, associa la stima del costo e il primo passo del percorso per raggiungerla. Periodicamente il nodo aggiorna i costi per raggiungere i nodi adiacenti e comunica la propria tabella ai vicini. Dopo alcuni scambi di informazioni, ciascun nodo avrà una tabella con una riga per ogni altro nodo della rete. Nell'utilizzo di questo tipo di algoritmi possono verificarsi problemi di *count-to-infinity* e la formazione di *cicli*, cui bisogna prestare molta attenzione. Esempi di protocolli distance vector sono il RIP e il DSDV.

- *Link State routing*: ogni nodo della rete acquisisce informazioni sullo stato dei collegamenti adiacenti ed inoltra queste informazioni a tutti gli altri nodi della rete tramite un "pacchetto link state". La ricezione di questi pacchetti permette ai nodi di costruire una mappa completa e aggiornata della rete, e di determinare il cammino minimo per raggiungere ogni altro nodo, ponendosi come radice dell'albero dei cammini minimi. Al termine dell'elaborazione è possibile costruire la tabella di routing memorizzando per ciascuna destinazione il nodo successivo sul cammino a costo minimo. Tra i vantaggi di questo algoritmo vi è la possibilità di gestire reti complesse formate da molti nodi, la capacità di convergere rapidamente al cammino minimo e la bassa probabilità che si generino cicli. Esempio di protocolli basati su Link State sono l'OSPF e l'OLSR.

2.1 PROTOCOLLI DI ROUTING PER RETI ACUSTICHE SOTTOMARINE

Per quanto riguarda il routing le reti sottomarine non sono diverse da altri tipi di reti wireless. Gli scenari su cui vanno ad operare possono estendersi per diversi chilometri, rendendo impossibili le comunicazioni single-hop. Inoltre le caratteristiche di propagazione sottomarine rendono molto più convenienti topologie multihop per via dei loro benefici: evitano i rilevanti fenomeni di attenuazione e dispersione che aumentano esponenzialmente con la distanza, riducono la potenza necessaria a coprire un hop, permettono di eseguire trasmissioni a frequenza e larghezza di banda maggiori ovvero a bit rate più elevati, e consentono di ottimizzare il consumo energetico.
L'approccio multihop, oltre ai vantaggi appena citati ha anche due principali svantaggi: da un lato, più nodi devono partecipare all'inoltro dei pacchetti, i quali vengono di conseguenza riprodotti più volte e costituiscono traffico aggiuntivo; d'altra parte, è necessario un meccanismo per la scelta del percorso e dei nodi da coinvolgere, il quale richiede overhead e ulteriori scambi di messaggi.
In ogni caso le problematiche che il routing sottomarino deve affrontare sono molte: per esempio la fisica di propagazione può portare a collegamenti dalle prestazioni che variano rapidamente nel tempo, rendendo difficile la scelta dei nodi da coinvolgere. Inoltre alcuni collegamenti possono esistere in una sola direzione, causando una sostanziale mancanza di equilibrio nei percorsi di routing.
Alla luce di queste osservazioni risulta facile intuire che un buon protocollo per reti sottomarine deve prevedere una quantità di messaggistica limitata, i protocolli proattivi quindi non sono molto adatti a questi scenari.
La maggior parte degli approcci esistenti in letteratura, che vedremo nel seguito, sono derivati dalle comunicazioni radio terrestri e non progettati direttamente per il canale sottomarino.
Sono necessari pertanto diversi passi in questa direzione, a partire da una più stretta corrispondenza tra politiche di routing e propagazione sottomarina, tenendo in considerazione quindi fenomeni quali la rifrazione, i canali di superficie, la convergenza e le zone d'ombra ecc.. Sarebbe interessante, ad esempio, fare in modo che i protocolli di routing riconoscano il verificarsi di questi eventi e si adattino ad essi migliorando le prestazioni di comunicazione.

2.2 ROUTING IN RETI TOLLERANTI AL RITARDO - DTN

Le Delay Tolerant Networks sono caratterizzate dalla frequente perdita di connettività dovuta alla scarsità e alla mobilità dei nodi, ai lunghi e variabili ritardi di propagazione, e agli alti tassi di errore sul bit. In questi ambienti, i popolari protocolli di routing ad hoc come AODV e DSR non riescono a stabilire rotte stabili, perché trasmettono i dati solo dopo aver trovato un percorso completo tra sorgente e destinazione. Tuttavia quando questi percorsi non esistono, i protocolli di routing devono adottare un approccio "store-carry-forward", che sfrutta la mobilità dei nodi per instradare i dati. Secondo questa filosofia, i dati vengono trasferiti dalla sorgente al primo nodo disponibile dove vengono salvati in attesa di un'opportunità per trasmetterli; i nodi mobili possono quindi trasportare i dati memorizzati in giro e cercare di trasmetterli mentre si muovono verso la destinazione. In questo modo i pacchetti vengono gradualmente spostati, immagazzinati e trasmessi attraverso la rete per raggiungere la destinazione con una certa probabilità. La tecnica più comune per massimizzare questa probabilità è di replicare il pacchetto a diversi nodi, in modo che almeno una copia possa raggiungere la destinazione.

Ci sono un certo numero di fattori da tenere in considerazione per il routing in una DTN. Uno di questi è la disponibilità o meno di informazioni sui contatti futuri. Ad esempio nelle comunicazioni interplanetarie (uno dei primi scenari per i protocolli di DTN) un corpo celeste può essere la causa di una interruzione di contatto, mentre la grande distanza è la causa dei ritardi di comunicazione. Tuttavia è possibile prevedere il verificarsi di contatti in futuro e la loro durata. Al contrario nelle reti disaster-recovery possono non esserci informazioni a priori sulla posizione dei dispositivi di comunicazione e sulla durata dei contatti, motivo per cui i contatti in questione sono definiti come intermittenti o opportunistici.
Un'altra importante considerazione riguarda la possibilità di sfruttare al meglio la mobilità dei nodi per il routing. Il livello di mobilità dei nodi nella rete può essere ricondotto a uno di questi tre casi:

1) Assenza di nodi mobili. In questo caso la presenza di contatti o meno dipende dalla qualità del canale di comunicazione, così come dall'ostruzione di eventuali oggetti in movimento.

2) Presenza di alcuni nodi mobili e di altri fissi. Quelli mobili sono sfruttati per la raccolta, lo stoccaggio, il trasporto e la trasmissione, dai nodi fissi alla destinazione. Assume particolare importanza il modo in cui i dati sono distribuiti tra essi.

3) Tutti (o quasi) i nodi della rete sono mobili. In questo caso un protocollo di routing ha più opzioni disponibili durante i contatti e può scegliere una particolare strategia.

Un altro fattore importante è la disponibilità di risorse di rete. Molti nodi sono limitati in termini di storage, rate di trasmissione, e durata della batteria. I protocolli di routing possono utilizzare queste informazioni per determinare come i dati debbano essere trasmessi e immagazzinati in modo da non sovra utilizzare le limitate risorse del canale.
Data la natura tipicamente delay tolerant di molte reti sottomarine, i protocolli di routing per DTN assumono importanza rilevante e meritano approfondimenti nel seguito.
Per creare una tassonomia dei protocolli di routing per DTN, ci si basa principalmente sulla creazione o meno di repliche di messaggi da parte del protocollo. I protocolli di routing che non replicano messaggi sono detti forwarding-based, viceversa quelli che li replicano sono detti replication-based.
Vi sono pro e contro per entrambi gli approcci, e quale dei due è preferibile sull'altro dipende principalmente dallo scenario di applicazione.
I protocolli basati sull'inoltro sono generalmente molto meno dispendiosi in termini di risorse, esistendo nella rete una sola copia del messaggio in un dato istante di tempo. Perciò quando la destinazione riceve la copia in questione, nessun altro nodo ne ha una. In questo modo, a differenza dell'approccio basato sulla replica, non c'è il bisogno di fornire un feedback al resto della rete (eccezione fatta per l'ack alla sorgente) affinché vengano cancellate le altre copie del messaggio. Tuttavia l'approccio forwarding-based di solito non è in grado di fornire un buon rapporto di consegna dei messaggi nella maggior parte degli scenari DTN, mentre il replication-based ottiene rapporti ben più alti grazie alle molte copie dei messaggi presenti nella rete.
Riassumendo il primo approccio impegna meno risorse a scapito della probabilità di una consegna corretta, mentre il secondo tende a impiegare più risorse dando però maggiori garanzie nella consegna. Inoltre, molti dei protocolli basati sul flooding non sono scalabili, alcuni di questi

come lo Spray and Wait tentano di trovare un compromesso limitando il numero di repliche di un dato messaggio.
E' importante notare che la maggior parte dei protocolli di routing DTN sono basati su euristiche e non ottimi; questo è dovuto al fatto che trovare algoritmi di routing e percorsi ottimi in una rete delay tolerant è, in generale, un problema NP-hard.

2.3 PRINCIPALI APPROCCI AL ROUTING SU RETI SOTTOMARINE NON DTN

Verranno ora descritti alcuni importanti approcci al routing underwater che si trovano in letteratura, partendo dai protocolli per reti non-DTN.

2.3.1 Routing basato sulla scoperta della topologia di rete

Il primo protocollo presentato è di tipo proattivo ed è in grado di stabilire autonomamente la topologia della rete, l'allocazione delle risorse, e il controllo del flusso di traffico. Tutto ciò grazie a un nodo che funge da manager di rete (nodo master). In particolare quest'ultimo, nel tentativo di stabilire percorsi efficienti per la consegna dei dati, ricostruisce la topologia della rete trasmettendo segnali di probe ai suoi vicini. Alla ricezione del primo probe i vicini vi aggiungono il loro id e lo inoltrano a loro volta ai propri vicini. Come mostrato in figura 4, viene così a crearsi un albero connesso, radicato nel nodo master. Quando il probe raggiunge i nodi di confine, dopo un timeout, essi rispondono al manager con un messaggio che notifica il completamento topologico, il quale ripercorre lo stesso percorso dei probe di scoperta. In questo modo quando gli arriva la notifica, il nodo master avrà tutte le informazioni per gestire molteplici sessioni di traffico attraverso la rete. Il protocollo affidandosi a un controller centralizzato è suscettibile di problemi di *single point of failure*. Inoltre dato il significativo scambio di messaggi per la scoperta topologica, non può essere adatto a reti sottomarine mobili e su larga scala.
Varianti all'approccio appena illustrato, prevedono algoritmi che ottimizzano la ricerca dei collegamenti e i relativi costi. Sussiste però la possibilità che non tutti i nodi-collegamenti siano

identificati, per esempio a causa di collisioni; da qui la necessità di configurazioni e regole che aiutino a ridurre sensibilmente questo rischio.

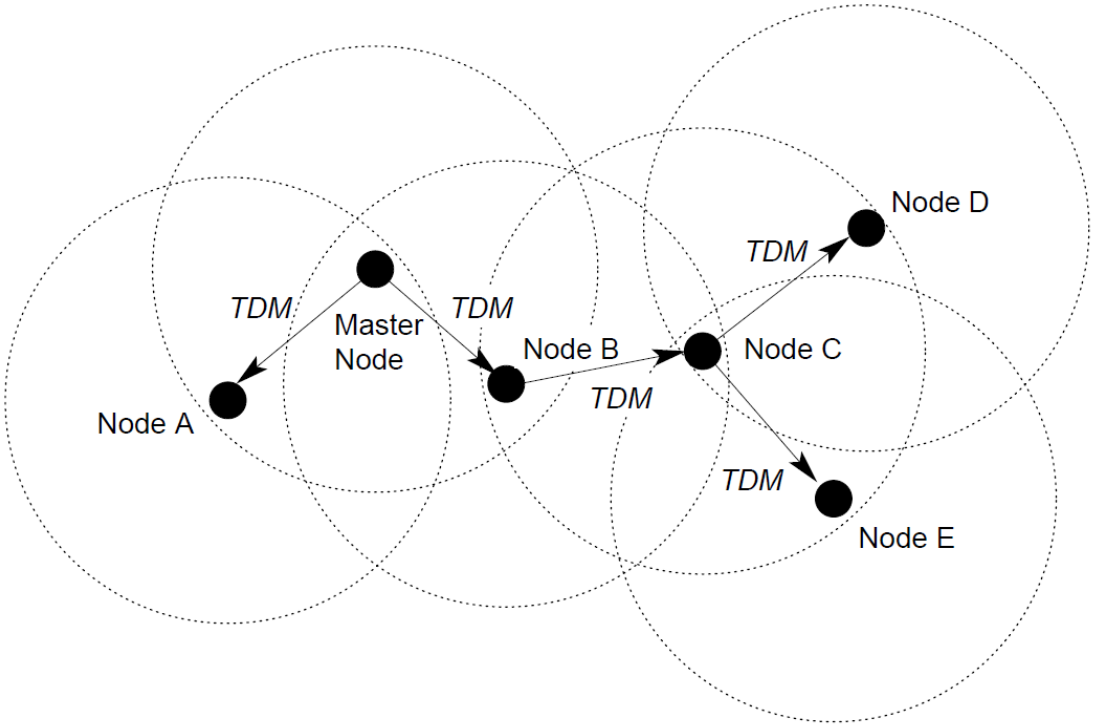


Fig. 4: Propagazione dei messaggi di scoperta topologica (TDM) dal nodo master ai nodi di confine

2.3.2 Vector Based Forwarding Routing

Il VBF (Vector Based Forwarding) è invece un protocollo di routing geografico dove i nodi prendono le decisioni di inoltro basandosi sulle informazioni disponibili in locale. Ogni pacchetto tiene traccia della posizione della sorgente S1, della destinazione S0 e del nodo cui viene inoltrato. Alla ricezione del pacchetto, il nodo calcola la sua posizione relativa misurando la distanza dal mittente e l'angolo di arrivo del segnale acustico: se si trova in una "pipe di routing" di un dato raggio W attorno al vettore S1S0 (come illustrato in figura 5), inoltra il pacchetto, altrimenti lo scarta.

Per ridurre ulteriormente il numero di pacchetti generati e di conseguenza il consumo di energia, è stato introdotto un "fattore di desiderabilità" che attiva solo un sottoinsieme dei nodi disponibili all'inoltro. Nel dettaglio, ad ogni ricezione un nodo determina se è abbastanza vicino

al vettore di routing, e in caso affermativo tiene il pacchetto per un periodo di tempo proporzionale a questo fattore. L'idea è di consentire ai nodi più "desiderabili" di inviare dati il prima possibile, facendo si che gli altri nodi debbano sopprimere la loro trasmissione, con conseguente diminuzione del numero di pacchetti duplicati nella rete. Le valutazioni del protocollo mostrano che scala molto bene col numero dei nodi, oltre ad essere robusto ed efficiente; tuttavia come tutti i protocolli geografici ha un costo, ovvero la conoscenza da parte dei nodi della propria posizione e di quella della destinazione. Ciò richiede tecniche di valutazione della posizione, che a loro volta richiedono hardware specifico o comunque l'implementazione di algoritmi di localizzazione (ad es. mediante nodi che, consapevoli della loro posizione, trasmettono periodicamente tale informazione consentendo agli altri di individuare la loro locazione tramite triangolazione).

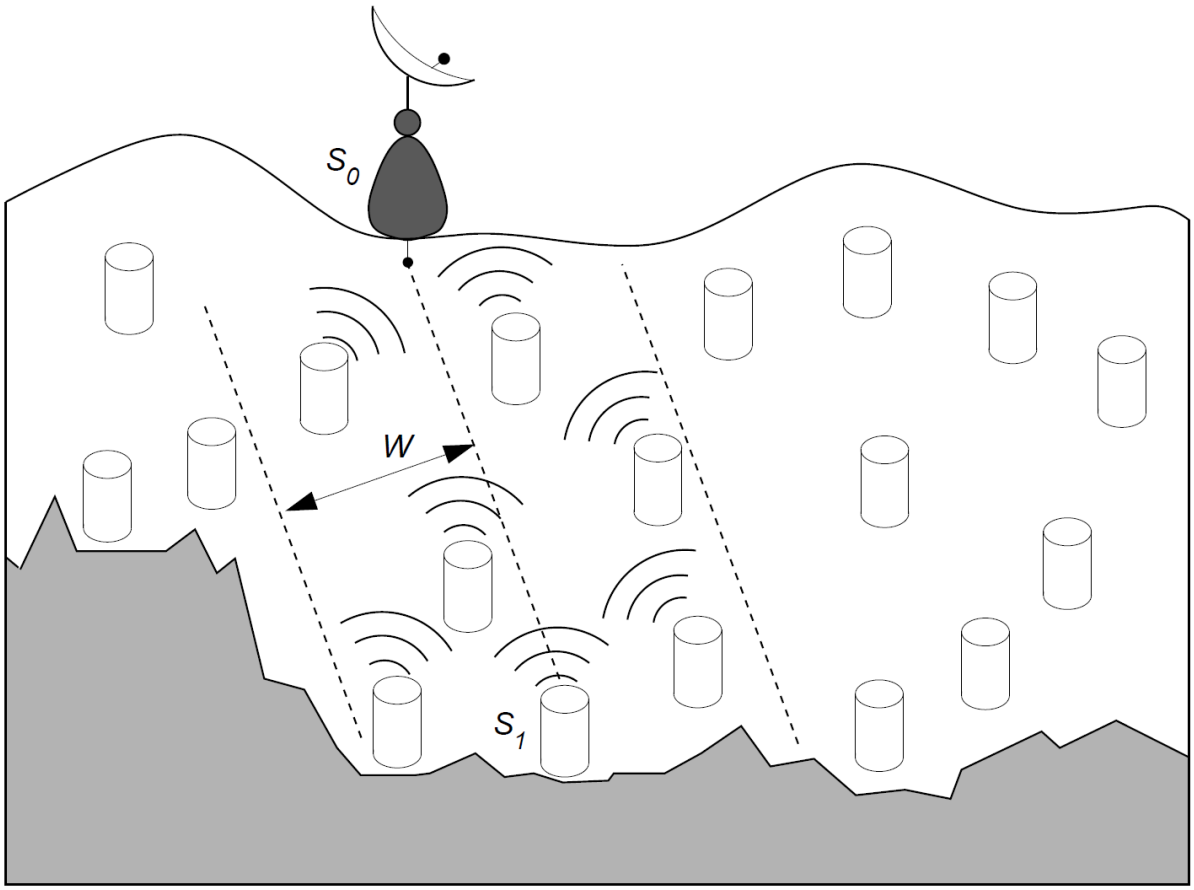


Fig. 5: schema di funzionamento del Vector-Based Forwarding

Un'estensione del VBF è l'HH-VBF (Hop-by-hop Vector Based Forwarding protocol) che mira a risolvere i problemi incontrati nel protocollo nativo dovuti all'utilizzo di un singolo vettore sorgente-sink. Questo vettore consente la creazione di una singola pipe virtuale tra sorgente e destinazione, che può incidere significativamente sull'efficienza del routing in quelle regioni di rete dove la densità dei nodi è bassa. Inoltre il VBF è troppo sensibile al raggio della pipe, rendendolo inadeguato nell'utilizzo pratico nelle reti. Consentendo la creazione di pipe virtuali hop dopo hop è possibile ridurre significativamente l'impatto di questi inconvenienti, aumentando il rapporto di consegna dei pacchetti e consumando meno energia.

2.3.3 Focused Beam Routing e Depth-based Routing

Un altro approccio proposto è l'FBR (Focused Beam Routing), un protocollo basato sulla conoscenza della posizione, che punta a minimizzare il consumo di energia per bit tramite un efficiente controllo del consumo di potenza. Tale protocollo è adatto per reti fisse e mobili, purché ogni nodo sia consapevole della sua posizione e di quella della destinazione.

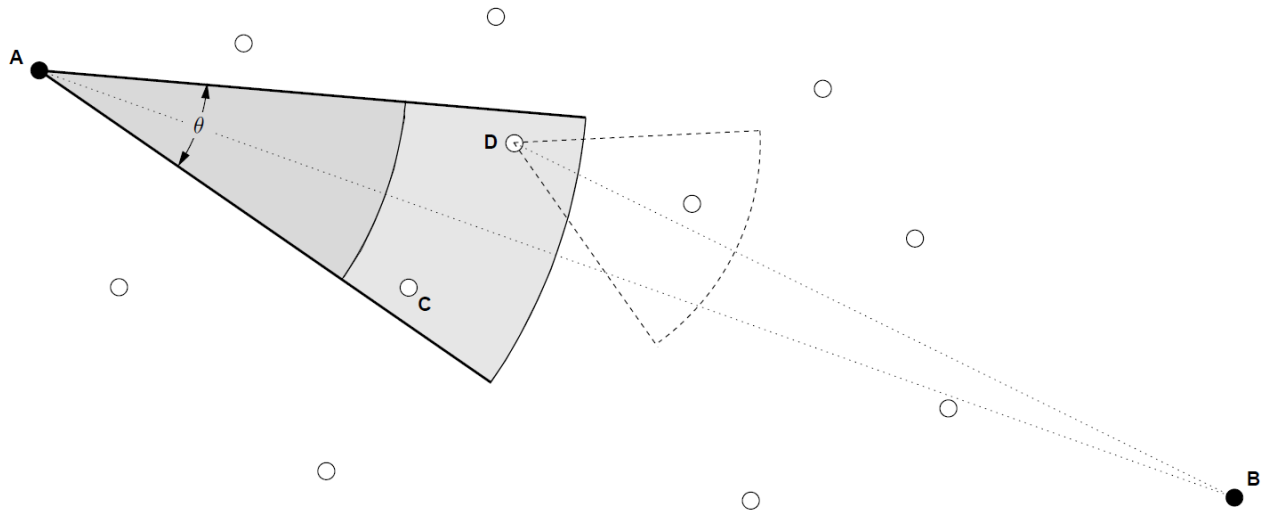


Fig. 6: Focused Beam Routing. I nodi all'interno del cono sono i candidati per il forward dei pacchetti

Il funzionamento viene illustrato facendo riferimento alla figura 6: il nodo A (sorgente) manda un RTS contenente la sua locazione e quella di B (destinazione), ai suoi vicini. Questo tipo di pacchetti viene inviato utilizzando la potenza trasmissiva più bassa, incrementata solo nel caso

nessun nodo venga raggiunto. Tutti i vicini di A che ricevono l'RTS calcolano la loro posizione rispetto al collegamento tra A e B, in modo da determinare se possono candidarsi all'inoltro del pacchetto. I nodi candidati per l'FBR sono quelli che risiedono nel cono mostrato in figura. Un candidato (ad esempio D) risponderà poi con un CTS, riceverà il pacchetto da A e ripeterà la medesima procedura andando a creare un percorso multihop fino alla destinazione B. Le simulazioni in questo caso hanno mostrato che accoppiando funzionalità MAC (rts / cts) a quelle di routing, si ottengono buone performance.

Le soluzioni proposte finora richiedono che i nodi sottomarini siano in grado di stimare la loro locazione al fine di eseguire il routing; l'approccio DBR (depth-based routing) prende invece decisioni basandosi solo sulla profondità dei nodi, evitando l'uso di complesse tecniche di localizzazione, dato che le informazioni sulla profondità possono essere facilmente estrapolate dai sensori di pressione di bordo.
Ogni nodo alla ricezione di un pacchetto calcola la profondità del nodo che gliel'ha inviato; nel caso risulti più vicino alla superficie si qualifica come candidato per inoltrare il pacchetto. In reti sottomarine dense di nodi, un algoritmo di questo tipo porta all'aumento notevole del consumo energetico, del numero di duplicati e quindi del rischio di collisioni. Meccanismi quali la soppressione di pacchetti ridondanti e timeout per la selezione del miglior next-hop tra i candidati, fanno fronte a questi problemi.

2.4 PANORAMICA SUI PROTOCOLLI DI ROUTING PER RETI DTN

2.4.1 Epidemic Routing

L'epidemic routing è basato sulle repliche continue dei messaggi contenute in ogni nodo, ad ogni contatto con qualche altro nodo, nel caso quest'ultimo non ne possieda già una copia.
Nel caso più semplice coincide col flooding, tuttavia esistono tecniche collaudate per limitare il numero di repliche. Un indice dei messaggi bufferizzati è mantenuto da ogni nodo, ed è la prima cosa che viene scambiata durante un incontro, per determinare quali pacchetti debbano essere inviati. Ogni messaggio ha un campo contenente il numero di hop, che oltre a determinare il

massimo numero di inoltri che può subire (un messaggio con hop count pari a 1 deve essere consegnato solo alla destinazione), limita l'utilizzo di risorse di rete.
Se l'hop count impostato nei messaggi e lo spazio disponibile nei nodi sono sufficientemente grandi, il messaggio sarà propagato all'intera rete, e il risultato sarà un elevato rapporto di consegna, a discapito però delle risorse utilizzate poiché l'algoritmo non prevede alcuna eliminazione dei messaggi replicati.

2.4.2 MaxProp

MaxProp nasce per il routing su reti veicolari, ed è anch'esso basato sul flooding. La sua peculiarità sta nell'essere in grado di determinare quali messaggi devono essere trasmessi per primi e quali devono essere scartati quando il buffer del nodo è pieno. A tal fine mantiene una coda ordinata in base alla destinazione di ogni messaggio, e in base alla probabilità stimata di trovare un percorso per quella destinazione. Quando due nodi si incontrano per prima cosa si scambiano i vettori con le stime delle probabilità di incontro con gli altri nodi. A partire da questi è possibile calcolare la route più breve, pesando i vari percorsi con la probabilità che un determinato collegamento non si verifichi. Il percorso col peso totale minore sarà selezionato per una particolare destinazione, e il suo costo verrà salvato. I messaggi sono quindi ordinati in base al costo per raggiungere la destinazione, e trasmessi (o scartati nel caso il buffer sia pieno).

L'algoritmo integra molti meccanismi, ognuno dei quali aiuta ad aumentare il rapporto di consegna dei messaggi e a ridurre i ritardi:

- l'acknowledgement da parte della destinazione che riceve correttamente un messaggio, permette di cancellare le copie extra del messaggio nei buffer dei vari nodi.
- ai messaggi con hop-count più basso viene data maggiore priorità, in tal modo si promuove una iniziale replica rapida.
- ogni nodo mantiene una lista degli hop che ha visitato in precedenza, in modo da evitare che rivisiti lo stesso nodo.

2.4.3 Spray and Wait

Lo Spray and Wait è un protocollo di routing DTN che cerca di beneficiare dei buoni rapporti di consegna degli approcci basati sulle repliche, con un basso utilizzo di risorse tipico del routing basato sul forward. Alla base della sua efficienza vi è l'assegnazione ad ogni messaggio generato, di un limite superiore *L* al numero di copie consentite nella rete.
Nella fase di Spray la sorgente trasferisce una copia a *L* nodi distinti. Quando un nodo riceve una copia entra nella fase di Wait: salva la copia in questione e attende finché non incontra la destinazione finale del messaggio.
L'algoritmo si presenta in due declinazioni rispettivamente note come Vanilla e Binary, che si differenziano nel meccanismo di "spray" delle *L* copie del messaggio. Vanilla è l'implementazione più semplice (quella descritta poc'anzi), dove una singola copia viene trasmessa ai primi L nodi distinti che vengono incontrati dopo la generazione del messaggio. La versione Binary invece trasmette L/2 copie al primo nodo incontrato; ognuno dei nodi trasmette poi metà delle proprie copie ai nodi che incontrerà che ancora non ne possiedono e così via. Quando un nodo cede tutte le sue copie tranne una, passa nella fase di Wait in attesa di un'eventuale trasmissione diretta alla destinazione finale. Il vantaggio di quest'ultima versione dell'algoritmo è che le copie sono disseminate nella rete molto più velocemente, e in fase di simulazione questo è confermato dai minori ritardi di consegna dei pacchetti.

2.4.4 Rapid

L'acronimo Rapid sta per protocollo di allocazione intenzionale delle risorse per DTN. L'obiettivo di questo algoritmo è di ottimizzare intenzionalmente una singola metrica di routing, tra il ritardo medio, il numero di ricezioni perse e il massimo ritardo. Il core di Rapid è basato sull'assegnazione di un valore di utilità a seconda della metrica da ottimizzare. L'utilità è definita come il contributo alla metrica, atteso dal pacchetto. Rapid replica prima quei pacchetti che risultano incrementare maggiormente la funzione di utilità; anch'esso è basato sul flooding e in quanto tale, la replicazione è soggetta alla disponibilità di storage e banda.

Il protocollo generale comprende le seguenti fasi:

- scambio di metadati per la stima dell'utilità del pacchetto;
- consegna diretta dei pacchetti destinati vicino alla destinazione;
- replica dei pacchetti in base all'utilità marginale;
- conclusione, quando i contatti si interrompono o tutti i pacchetti sono stati replicati.

CAPITOLO 3

P-SPRAY ROUTING PROTOCOL

Nel seguente capitolo viene presentato nel dettaglio il protocollo oggetto di questa tesi.

3.1 CRITERI DI PROGETTO

Il protocollo è progettato per reti di nodi sottomarini mobili, ignari delle proprietà topologiche della rete e della posizione dei nodi.
L'algoritmo è di tipo *Hop-by-Hop,* i pacchetti infatti non contengono alcuna informazione di routing e ogni nodo decide autonomamente come instradare i dati (si veda a tal proposito il capitolo 2), ed è basato sulla generazione di repliche ottimizzata dal meccanismo di binary spray (presentato nella sezione 2.4.3) per migliorare i ritardi di consegna, ed evitare un eccessivo spreco di risorse.
Ad ogni pacchetto generato viene associata una deadline raggiunta la quale il pacchetto viene scartato, e un numero massimo di repliche, che ad ogni forwarding viene dimezzato e una volta raggiunto il valore 1 consente l'invio del pacchetto alla sola destinazione.
Presuppone la conoscenza a priori della statistica degli incontri tra i nodi: le repliche dei pacchetti sono inviate in base alla probabilità che ogni nodo ha di incontrare la destinazione, condizionata al tempo trascorso dall'ultimo incontro con essa (la P nel nome del protocollo richiama il concetto di probabilità).

3.2 PANORAMICA SUL PROTOCOLLO

3.2.1 Simulazione della statistica dei tempi di incontro in reti mobili sottomarine

La statistica degli incontri è stata ottenuta da uno script scritto nel linguaggio Matlab, a partire da un file contenente le sole coordinate dei nodi, generato dal network simulator come output di alcune simulazioni in questo scenario:

numero nodi -> 10
modello di mobilità -> Gauss-Markov 3D, con $\alpha = 0.8$, pitch = 0.02, velocità = 6 m/s,

area	->	9x5x1 km^3
durata simulazione	->	6 mesi
rilevazione coordinate	->	ogni 60 secondi

Lo script in Matlab opera secondo i seguenti passi:

1. A partire dalle coordinate calcola la distanza tra ogni coppia di nodi ad intervalli di 60 secondi.
2. Tramite le formule empiriche del canale sottomarino (illustrate nel capitolo 1), calcola l'attenuazione riferita alle distanze ottenute in precedenza, e il rumore (entrambi in dB), ponendo la frequenza f = 24 KHz, l'esponente di path-loss k = 1.75, s = 0.5 e w = 0.
3. Calcola l'SNR in ricezione tra ogni coppia di nodi ai vari istanti di tempo, fissando la potenza di trasmissione a 150 dB e sottraendo ad essa le componenti di attenuazione e rumore; nel caso il risultato dell'operazione sia una quantità maggiore di una soglia fissata a 15 dB, significa che i nodi riescono a comunicare.
4. Sfruttando i dati sulla connettività così ricavati, calcola le durate dei contatti (intervallo di tempo durante il quale due nodi riescono a comunicare) e degli inter-contatti (intervallo di tempo tra 2 contatti) tra ogni coppia di nodi, e il loro numero totale.
5. Sottraendo intervalli di tempo da 10 minuti ciascuno alle durate degli inter-contatti, ne effettua una suddivisione in base alla loro durata residua, quantificata sempre in termini di intervalli da 10 minuti. Ricava quindi la statistica (pdf) degli inter-contatti che terminano in un dato intervallo, condizionata al fatto che sono già iniziati da X minuti, con X multiplo di 10.
6. Calcola la funzione di distribuzione cumulativa (cdf) sommando le precedenti statistiche; viene così determinata la probabilità che un inter-contatto finisca entro un dato tempo (multiplo di 10 minuti) condizionata al fatto che l'inter-contatto è già iniziato da un certo tempo (sempre multiplo di 10 minuti).

Più nel dettaglio:

1. carica il file con le coordinate dei nodi rilevate a intervalli di 60 secondi;
2. crea una matrice 3D con le coordinate di tutti i nodi in funzione del tempo trascorso;
3. crea una matrice 3D con le distanze di ogni nodo da tutti gli altri in funzione del tempo trascorso;
4. calcola l'assorbimento con la formula di Thorp per f = 24 kHz;
5. crea una matrice 3D delle attenuazioni a partire da quella delle distanze, applicando la formula di Thorp con esponente di path-loss pari a 1.75;
6. calcola il rumore ponendo s = 0.5 e w = 0;
7. fissa la potenza trasmissiva a 150 dB e la soglia di SNR in ricezione a 15 dB;
8. crea una matrice 3D degli SNR a partire da quella delle attenuazioni, sottraendo alla potenza di trasmissione i valori di attenuazione e rumore;
9. crea una matrice 3D di booleani a partire dalla matrice degli SNR, controllando che gli SNR siano maggiori della soglia fissata in ricezione;
10. crea la matrice con le durate dei contatti e degli inter-contatti tra ogni coppia di nodi, contando gli 1 e gli 0 consecutivi della matrice di booleani precedente (se il valore è 1 i nodi sono in grado di comunicare ed è incrementata la durata di un contatto, se è 0 è incrementata la durata dell'inter-contatto) e moltiplicando per 60 ogni valore che si ottiene (essendo ciascun valore della matrice riferito a rilevazioni effettuate ogni minuto)
11. crea una matrice col numero di inter-contatti per ogni nodo;
12. crea una matrice 4D: itera su intervalli di 10 minuti, li sottrae alla durata di ciascun intercontatto, divide il risultato per 600 determinando da quanti intervalli di 10 minuti è composta la vita residua di ciascun inter-contatto, e usa questo numero come indice di colonna dove incrementare il valore; in ogni riga della matrice (cioè per ogni coppia di nodi) avremo così il numero di inter-contatti che finiscono tra 0 e 10 minuti nella prima colonna, tra 10 e 20 minuti nella seconda, e così via, considerando che ad essi abbiamo già sottratto una certa durata multipla di 10 minuti. Dividendo le quantità di ciascuna colonna per il numero totale di inter-contatti se ne ricava la pdf, e sommando i valori della pdf fino a un certo indice, si ricava la cdf, ovvero il numero totale di inter-contatti che finiscono *entro* un dato tempo.

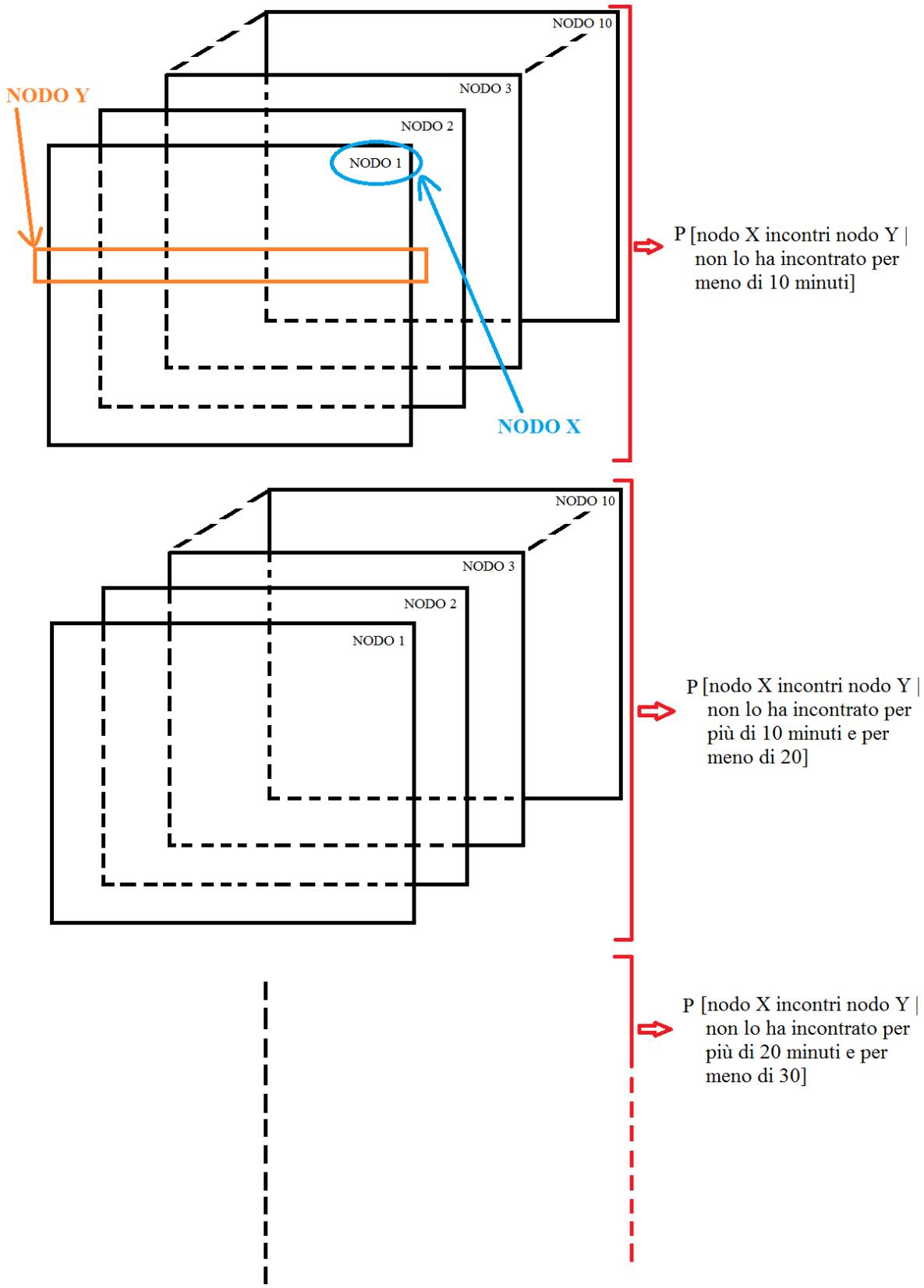
Il risultato dell'esecuzione dello script appena descritto è una matrice 4D di probabilità, la cui struttura è mostrata in figura 7. Ogni matrice 3D contiene le probabilità d'incontro di ciascun nodo con tutti gli altri: l'indice della terza dimensione identifica il nodo preso in considerazione, l'indice di riga indica quello che si vorrebbe incontrare, mentre l'indice di colonna scandisce gli intervalli da 10 minuti entro cui avviene l'incontro. La quarta dimensione è necessaria per tener traccia dello scorrere del tempo (in intervalli di 10 minuti), e quindi per condizionare le probabilità.

Per esempio se vogliamo conoscere la probabilità che il nodo 6 incontri il nodo 4 entro 20 minuti, dato che sono passati 25 minuti dal loro ultimo incontro, bisognerà accedere alla cella di indice [3][4][2][6] della struttura dati generata da Matlab:

3→ perché sono passati 25 minuti dal loro ultimo incontro, bisogna accedere quindi alla terza matrice 3D, ovvero quella che condiziona le probabilità al fatto che sono passati tra i 20 e i 30 minuti dall'ultimo incontro;
4 → ovvero l'indice di riga corrispondente a quello del nodo che voglio incontrare;
2 → è l'indice di colonna corrispondente alla probabilità d'incontro entro 20 minuti;
6 → è l'indice relativo al nodo in considerazione, ovvero quello della matrice 2D cui accedere.

Tutte le probabilità sono state poi salvate in un file .txt, da cui il protocollo ricostruisce l'array 4D in fase di esecuzione. Quando si accede alla nuova struttura dati, bisogna porre attenzione al fatto che gli indici vanno scalati tutti di uno, in quanto la numerazione nella maggior parte dei linguaggi di programmazione (compreso il linguaggio c++ utilizzato dal simulatore) parte da 0, differentemente da Matlab dove parte da 1.

<u>Osservazione:</u>
la lunga durata della simulazione e l'alta velocità dei nodi (6 m/s), sono giustificate dalla necessità di ottenere un alto numero di contatti, tale da estrapolare una statistica abbastanza accurata.


NODO 10
NODO 3
NODO 2
NODO 1
NODO Y
NODO X
P [nodo X incontri nodo Y | non lo ha incontrato per meno di 10 minuti]
NODO 10
NODO 3
NODO 2
NODO 1
P [nodo X incontri nodo Y | non lo ha incontrato per più di 10 minuti e per meno di 20]
P [nodo X incontri nodo Y | non lo ha incontrato per più di 20 minuti e per meno di 30]

3.2.2 I pacchetti

E' previsto l'utilizzo di 4 header da aggiungere ai vari pacchetti, creati ad-hoc per adempiere ai compiti previsti dal protocollo:

1. "Hello Header" → usato per notificare a chi riceve il pacchetto che lo contiene, la presenza del mittente nel suo raggio di copertura. La dimensione è di 1 byte.

2. "Packets To Send" (PTS) Header → formato da un array di 10 elementi, contenente la deadline massima dei pacchetti in coda destinati a ciascun nodo, identificato dall'indice dell'array + 1. Le deadline sono espresse mediante unsigned int da 32 bit, la dimensione dell'header è quindi 4*10 byte + 1 byte per il char identificativo = 41 byte.

Deadline massima dei pacchetti in coda per il nodo 1
Deadline massima dei pacchetti in coda per il nodo 2
Deadline massima dei pacchetti in coda per il nodo 3
Deadline massima dei pacchetti in coda per il nodo 4
…
…
…
Deadline massima dei pacchetti in coda per il nodo 10

3. "Packets I Can Receive" (PCR) Header → formato da un array 10 di elementi contenente le deadline minime che i pacchetti devono avere affinché possano essere replicati a un altro nodo. La struttura è la stessa dell'header PTS, quindi anche le dimensioni sono invariate, così come il modo in cui vengono identificati i nodi.

Deadline minima che devono avere i pacchetti destinati al nodo 1 affinché possano essere inviati
Deadline minima che devono avere i pacchetti destinati al nodo 2 affinché possano essere inviati
Deadline minima che devono avere i pacchetti destinati al nodo 3 affinché possano essere inviati
…
…
…
…
Deadline minima che devono avere i pacchetti destinati al nodo 10 affinché possano essere inviati

4. “Data Header” → aggiunto a ogni pacchetto dati per tener traccia del suo expiration time, e del numero di copie ancora consentite dal binary spray. L’informazione temporale è espressa mediante un int da 32 bit per, mentre il numero di copie è espresso da 1 byte. La dimensione dell’header è quindi di 6 byte (4 dell'expiration time + 1 del numero di copie + 1 del char identificativo).

In ogni pacchetto, oltre all’eventuale presenza di uno di questi header, è costante la presenza dell’header IP, necessario per le trasmissioni, ed utilizzato in diverse circostanze per ricavare gli indici dei nodi in gioco.
In realtà l’utilizzo di IP non è ottimale come scelta per le reti sottomarine. Tipicamente le UAN sono formate da pochi nodi ed è sufficiente qualche bit per indirizzarli tutti. I 32 bit degli indirizzi Ipv4 rappresentano quindi overhead per buona parte, ma il loro utilizzo si rende necessario per questioni legate alle simulazioni: la maggior parte delle applicazioni del simulatore si basa infatti su IP, e le modifiche da apportare sono ingenti nel caso per esempio si vogliano utilizzare indirizzi a 8 bit. Inoltre molti dei campi messi a disposizione dall'header IP non sono necessari: nel caso di P-Spray vengono trasmessi solo gli indirizzi di sorgente e destinazione, e il TTL.
.

3.2.3 L’algoritmo

A supporto del protocollo, oltre alla tabella delle probabilità d'incontro e agli header in precedenza descritti, vi sono le seguenti strutture ed elementi:

- 4 timer: waitPTS, waitPCR e waitDATA, che scandiscono il tempo durante il quale i nodi aspettano i pacchetti in questione, e HelloTimer che gestisce gli intervalli di invio degli Hello Packet; i primi 3 sono settati a 5 secondi, un intervallo ritenuto ragionevole considerando la massima distanza che 2 nodi possono avere nell’area di interesse (9x5x1 Km) e la velocità di propagazione del suono (1500 m/s). HelloTimer invece ha una durata casuale tra 0 e 80 secondi, impostata di volta in volta onde evitare collisioni, dovute all’invio contemporaneo di HelloPacket da parte di più nodi. Alla scadenza del waitPTS, se il nodo non ha nessun pacchetto PTS nella coda dedicata, resetta l'HelloTimer e ricomincia la procedura di routing

con l'invio di un Hello Packet; la stessa cosa avviene alla scadenza dei timer waitPCR e waitDATA.

- Coda dove salvare i pacchetti generati dalle applicazioni e le copie ricevute dagli altri nodi, fino a un massimo di 1000 pacchetti.

- Coda dove salvare tutti i pacchetti PTS ricevuti entro la scadenza del timer waitPTS. La sua dimensione è pari a 9 (il numero max di pacchetti PTS che è possibile ricevere in una rete di 10nodi), ed è necessaria per selezionare il nodo con cui intraprendere la comunicazione.

- Array di 10 elementi, la cui cella di indice *i* contiene l'istante di tempo dell'ultimo incontro col nodo di indice *i*+1; il valore contenuto nelle rispettive celle viene aggiornato alla ricezione di ogni pacchetto, ed è necessario per estrarre le probabilità corrispondenti dalla tabella.

L'algoritmo viene di seguito illustrato facendo riferimento alla figura 8. Si suppone che il nodo A generi pacchetti destinati ai nodi B e C, e che incontri per primo il nodo B. L'evolversi degli eventi segue questa sequenza:

1) A, B e C fanno partire l'HelloTimer, scaduto il quale mandano l'*Hello Packet* per cercare nodi con cui scambiare informazioni. Passano quindi nello stato di attesa di pacchetti PTS, con la partenza del relativo timer (*waitPTS*).

2) A genera pacchetti destinati a B e C, i quali vengono accodati impostandone ad esempio, le deadline dopo 1 ora (il tempo di vita fissato per ogni pacchetto), e il numero massimo di repliche consentite a 10. Contestualmente viene aggiornato l'header PTS con le deadline massime dei pacchetti.

3) Supponiamo che B sia il primo nodo ad inviare l'*Hello Packet*, ricevuto sia da A che da C. Questi ultimi cancellano il proprio HelloTimer, mandano in risposta il proprio pacchetto

PTS e passano nello stato di attesa del pacchetto PCR, facendo partire il relativo timer (*waitPCR*).

4) B riceve i PTS sia di A che di C, e li mette nella coda ad essi dedicata. Alla scadenza del timer waitPTS controlla i pacchetti che ha ricevuto, e sceglie il nodo con cui comunicare in base al numero di pacchetti che ha per esso (in caso di uguaglianze la scelta ricade sul nodo il cui PTS è stato ricevuto per primo). Nel caso di figura 8, C non ha generato pacchetti per nessuno, mentre A ha pacchetti per B (oltre che per C) venendo così scelto per iniziare lo scambio.

   Inizia così la creazione dell'header PCR consultando la tabella delle probabilità. Lo scopo è quello di fornire al nodo A, un parametro per l'eventuale forward di pacchetti destinati agli altri nodi (C nel nostro caso). Il parametro in questione è il tempo entro il quale B ha una buona probabilità di incontrare C, che si traduce nella minima vita residua che devono avere i pacchetti di A destinati a C, per essere replicati a B.

   Per ottenere questa informazione B deve accedere all'array 4D col meccanismo spiegato precedentemente; in particolare il tempo con cui condizionare la ricerca viene dato dall'array degli istanti dall'ultimo incontro, mentre la deadline max trasmessa nel pacchetto PTS fissa il limite alle colonne da controllare. L'operazione si svolge dunque tramite un ciclo sull'indice di colonna di una riga ben precisa della tabella, e si interrompe quando viene trovato un valore superiore a una certa soglia fissata a priori. A questo punto è possibile predire l'intervallo di tempo entro cui B presumibilmente incontrerà C, così da scriverlo nella corrispondente cella dell'header del pacchetto PCR.

5) Al termine della scansione della tabella delle probabilità, l'header PCR completo viene aggiunto a un pacchetto e inviato da B in risposta al nodo A. Parte quindi il timer *waitData,* che scandisce l'attesa dei pacchetti dati da A.

6) Il nodo A una volta ricevuto il pacchetto PCR, trasmette subito i pacchetti destinati a B, e successivamente una copia di quelli destinati a C la cui deadline è maggiore di quella contenuta nell'header PCR (ovvero quei pacchetti che hanno una vita residua tale da avere buone probabilità di arrivare a destinazione tramite B), e il cui numero di copie residue è

maggiore di 1 (quando è 1 il binary spray prevede che il pacchetto sia inviato solo alla destinazione). Prima di eventuali repliche il valore delle copie residue viene dimezzato.

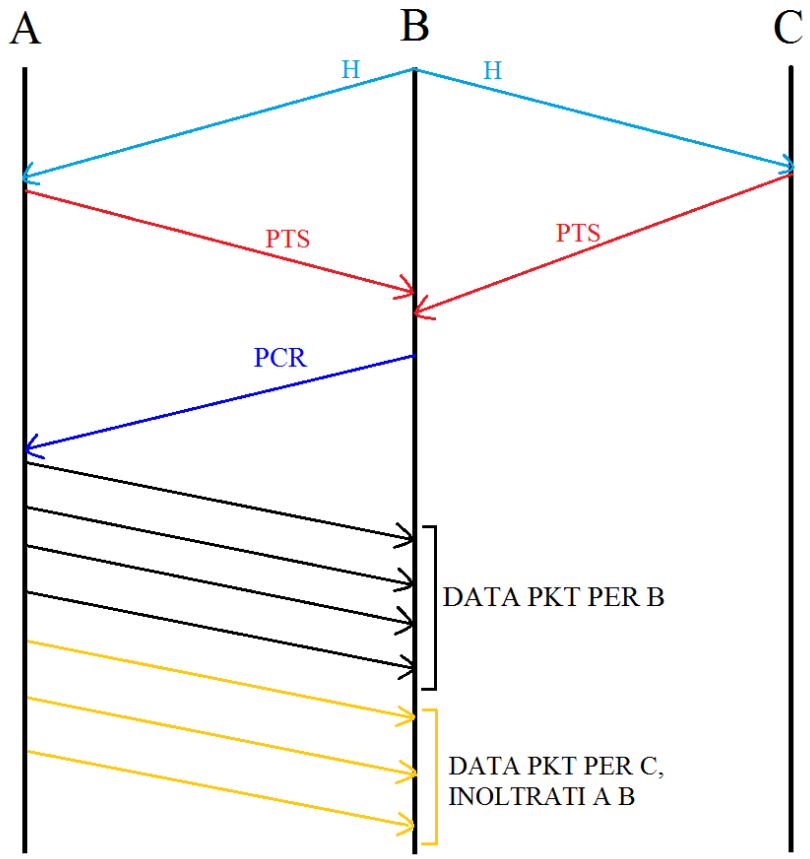


Fig. 8: Esempio di funzionamento del protocollo

Osservazioni:

1) Il protocollo non prevede meccanismi di acknowledgement e i pacchetti vengono cancellati dalle code dei vari nodi una volta scaduta la loro deadline o quando sono consegnati alla destinazione; la scelta è giustificata dalla necessità di contenere l'overhead, e di massimizzare le trasmissioni: l'eventuale attesa di ack dopo ogni invio, considerando i tempi di propagazione negli scenari underwater, ridurrebbe notevolmente il tempo disponibile per effettuare ulteriori trasmissioni.

2) Il meccanismo di forwarding del protocollo prevede l'invio delle copie dei pacchetti scorrendo la coda dalla fine (strategia Last In First Out - LIFO), partendo cioè dai pacchetti accodati per ultimi. Questo per evitare che siano replicati sempre gli stessi pacchetti (quelli all'inizio della coda), e che gli ultimi ad essere generati giungano a destinazione con scarsa probabilità.

Pseudocodice per un generico nodo:

```
inizializza HelloTimer;
if (HelloTimer scade)
   { manda hello packet;
     inizializza timer waitPTS(stato nodo = waitPTS); }

if(genera pacchetti)
   {accoda (pacchetti);
    calcola deadline massima(pacchetti);
    aggiorna header PTS (deadline massima);
    imposta numero massimo copie(pacchetti); }

ricevi pacchetto
{
    salva istante di tempo della ricezione nell'array con le info sui tempi dall'ultimo incontro;
    if (pacchetto ricevuto = hello && HelloTimer non è scaduto)
       {cancella HelloTimer;
        manda pacchetto PTS;
        inizializza timer waitPCR (stato nodo = waitPCR);}
    elseif (pacchetto ricevuto = PTS &&  stato nodo = waitPTS)
       accoda pacchetto ricevuto nella codaPTS;
    elseif (pacchetto ricevuto = PCR  &&  stato nodo = waitPCR)
       {manda pacchetti destinati al nodo con cui sto comunicando;
        imposta numero di copie residue dei pacchetti in procinto di essere replicati;
        manda replica dei pacchetti per gli altri nodi con deadline > del valore nell'header PCR;}
    elseif (pacchetto ricevuto = pacchetto dati per me  &&  stato nodo = waitDATA)
       {consegna pacchetto ai livelli superiori;
        resetta timer waitData (stato nodo = waitDATA); }
```

```
    elseif (pacchetto ricevuto = pacchetto dati per un altro nodo  &&  stato nodo = waitDATA)
        {mette in coda il pacchetto con la deadline originale;
         resetta timer waitData (stato nodo = waitDATA); }
    else
         scarta pacchetto
}


if  (timer waitPTS scade && codaPTS ha elementi)
    {controlla i pacchetti PTS in coda;
     scegli il nodo con cui comunicare in base a chi ha più pacchetti destinati a lui;
     crea header PCR consultando il contenuto dell'header PTS  e la tabella delle probabilità;
     manda pacchetto PCR al nodo scelto  (stato nodo = waitDATA);}
else
     resetta HelloTimer;

if (waitPCR timer scade || waitDATA timer scade)
     resetta HelloTimer;
```

CAPITOLO 4

VALUTAZIONE DELLE PRESTAZIONI

4.1 SIMULATORI DI RETE

Per implementare e testare il protocollo si è reso necessario l'utilizzo di un simulatore di rete con la capacità di simulare il canale sottomarino. Inizialmente la scelta era ricaduta su NS3, ma il mancato supporto al routing underwater e le crescenti difficoltà in sede di adattamento, hanno orientato la scelta definitiva sul predecessore NS2 unitamente alle librerie NS-Miracle.

4.1.1 Network Simulator 2 (NS2)

Ns è un simulatore ad eventi discreti destinato alla ricerca, che fornisce un buon supporto per molti tipi di protocolli sia su reti cablate che wireless. Nasce nel 1989 come variante del simulatore di reti REAL, e si è evoluto sostanzialmente col passare degli anni. Nel 1995 il suo sviluppo è stato supportato dalla DARPA, dalla Xerox PARC, dalla UCB e dall' USC / ISI. Attualmente invece è supportato ancora dalla DARPA col progetto SAMAN e dalla NSF col progetto CONSER, entrambe in collaborazione con altri gruppi di ricercatori come l' ACIRI.
Nonostante i molti anni di utilizzo e l'incremento continuo del numero di sviluppatori, ns2 è tutt'altro che un prodotto finito, soggetto a un continuo lavoro di ricerca e sviluppo, oltre che alla correzione di numerosi bug.
I linguaggi di programmazione su cui è basato sono OTcl e C++. Da una parte OTcl è compilato a runtime e consente di impostare velocemente parametri e configurazioni di rete; dall'altra C++ dà allo sviluppatore la capacità di programmare i nodi gestendo algoritmi efficienti, header, strutture dati ecc.
Tra le sue caratteristiche principali, l'ampio e specifico supporto a molte tipologie di reti, l'elevata modularità e la presenza di diversi modelli di simulazione con relativa documentazione, fanno si che attualmente sia ancora molto impiegato; non mancano però alcuni aspetti negativi come la ridotta scalabilità e l'impossibilità di riutilizzare il codice in sistemi reali.

4.1.2 NS-Miracle

NS-Miracle (Network Simulator - Multi InteRfAce Cross Layer Extension) è un insieme di librerie sviluppate dal Dipartimento di Ingegneria dell'Informazione dell'Università di Padova per incrementare le funzionalità offerte da NS2. Come si può intuire dall'acronimo, fornisce un efficiente gestione dei messaggi cross-layer e supporta la coesistenza di moduli multipli in ciascuno strato dello stack protocollare. In uno stesso nodo ad esempio, possono essere utilizzati diversi indirizzi IP, collegamenti, livelli fisici e livelli MAC.
NS-Miracle facilita l'implementazione e la simulazione in NS2 di moderni sistemi di comunicazione, reti sottomarine comprese; inoltre, grazie alla sua modularità, il codice è facilmente portabile, riutilizzabile ed estensibile.

4.2 SCENARIO E PARAMETRI DEI TEST

La configurazione generale della rete utilizzata nei test del protocollo è la seguente:

- Area: 9 Km lunghezza, 5 Km larghezza, 1 Km profondità
- Numero di nodi: 10
- Modello mobilità: Gauss Markov 3D ($\alpha = 0.8$, pitch = 0.02, vel = 6 m/s)
- Durata Simulazioni: 1 giorno circa
- Dimensioni pacchetti: 125 byte
- Tempo di vita dei pacchetti: 1 ora
- Destinazione pacchetti: casualmente scelta tra gli altri nodi
- Numero massimo di repliche: 10 (= numero nodi)

Parametri di trasmissione e di canale:

- Frequenza portante: 24 KHz
- Larghezza di Banda: 5 KHz
- Potenza Trasmissione: 150 dB
- Bit Rate: 4800 bps
- Tecnica di Modulazione: BPSK
- Esponente di path-loss: 1.75
- Shipping factor: 0.5
- Wind Speed: 0

I seguenti moduli sono stati utilizzati per realizzare lo stack di rete completo:

1. Livello Canale: Modello di canale definito nel capitolo 1
2. Livello Fisico: BPSK
3. Livello DataLink: ALOHA + MLL
4. Livello Rete: IP + **P-SPRAY**
5. Livello Trasporto: UDP
6. Livello Applicazione : CBR

4.3 RISULTATI

Nel seguente paragrafo saranno presentati i risultati delle simulazioni, svolte nello scenario e con i parametri descritti in precedenza.

Le prestazioni del protocollo sono state valutate principalmente in termini di:

- *Packet Delivery Ratio (PDR)* = $\frac{\#\text{ pacchetti consegnati entro la deadline}}{\#\text{ pacchetti generati}}$

- *Replication ratio* = $\frac{\#\text{ repliche generate}}{\#\text{ pacchetti generati}}$

- *Delivery Delay* = ritardo da quando un pacchetto è generato a quando è consegnato

al variare di 2 parametri:

- λ = Data Generation Rate per Node (espresso in bps)

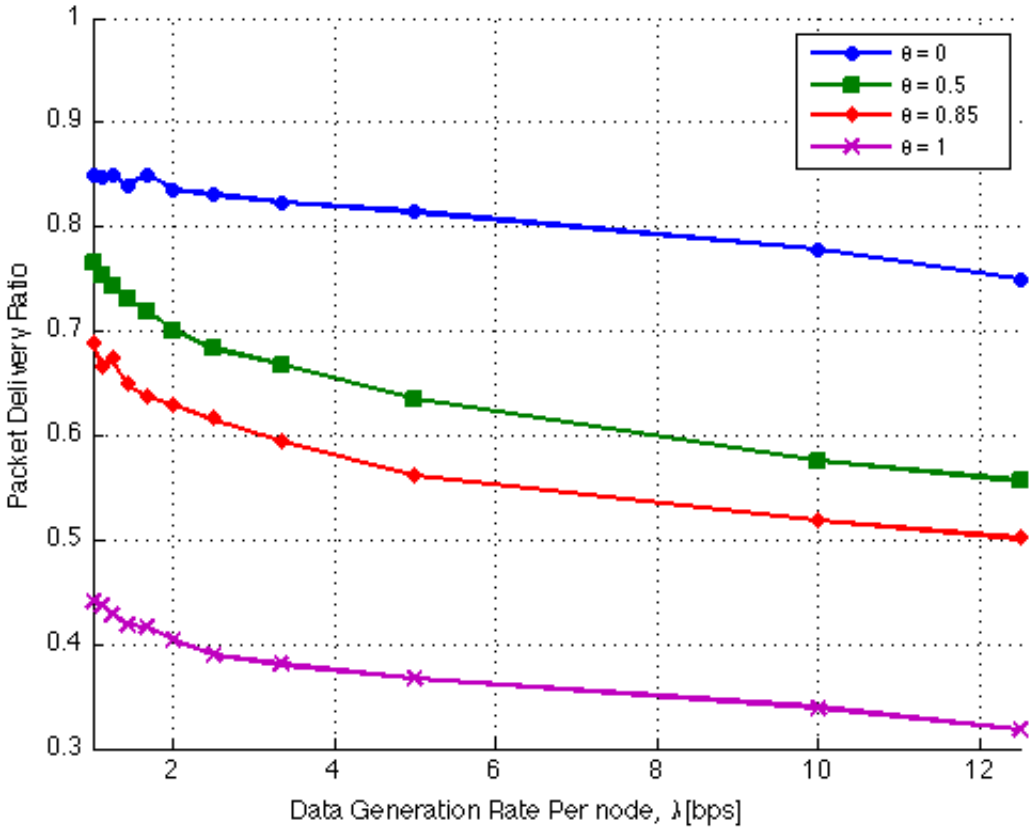
- θ = probabilità minima di incontro tra 2 nodi entro una deadline prefissata (usata per selezionare a quali nodi replicare i pacchetti)

Osservazione:

Per come è stato definito il protocollo, ai valori limite che θ può assumere corrispondono i seguenti comportamenti:

- $\theta = 0$ --> flooding
- $\theta = 1$ --> consegna consentita solo alla destinazione

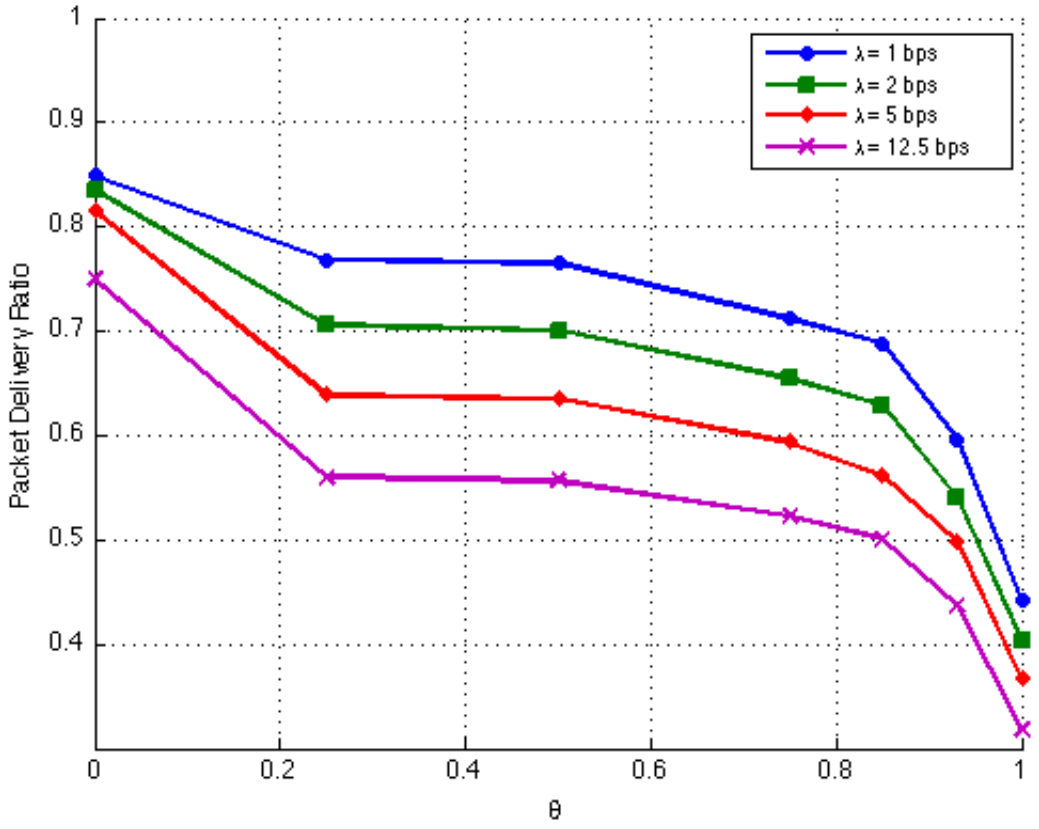
Grafico 1:



Il primo grafico mostra il PDR al crescere di λ, per diversi valori di Θ. I risultati rispettano le previsioni in quanto una soglia di probabilità più bassa implica vincoli meno stringenti alla possibilità di replica dei pacchetti (il requisito di vita residua che i pacchetti devono avere per essere replicati è più basso), e il conseguente alto numero di repliche fa si che aumenti il numero di pacchetti che arrivano a destinazione.

Il calo della percentuale di consegne al crescere di λ risulta invece fisiologico: una maggior quantità di dati nella rete porta in generale alla congestione; il gran numero di pacchetti nelle code fa si che parte di essi non venga processata durante gli incontri, i tempi necessari agli scambi infatti si allungano, e diversi pacchetti scadono o vengono persi.

Grafico 2:



Osservando il grafico 2, dove viene preso in considerazione il PDR in funzione di Θ per diversi valori di λ, il ragionamento non cambia: Θ è inversamente proporzionale al numero di repliche dei pacchetti nella rete, e quindi al numero di arrivi a destinazione; a conferma di ciò si noti che con $\Theta = 0$ raggiungiamo i PDR maggiori, che decrescono all'aumentare di Θ fino a raggiungere i valori più bassi con $\Theta = 1$.

Il controllo del numero di copie per ogni pacchetto mediante il binary spray, fa si che pur aumentando λ, a bassi valori di Θ (cui corrisponde la generazione di molte repliche) non vi siano abbassamenti del PDR dovuti all'incapacità di gestire il traffico.

In generale a valori più bassi di λ corrispondono PDR più alti, in quanto risulta più facile consegnare i pacchetti generati entro le rispettive deadline.

Grafico 3:

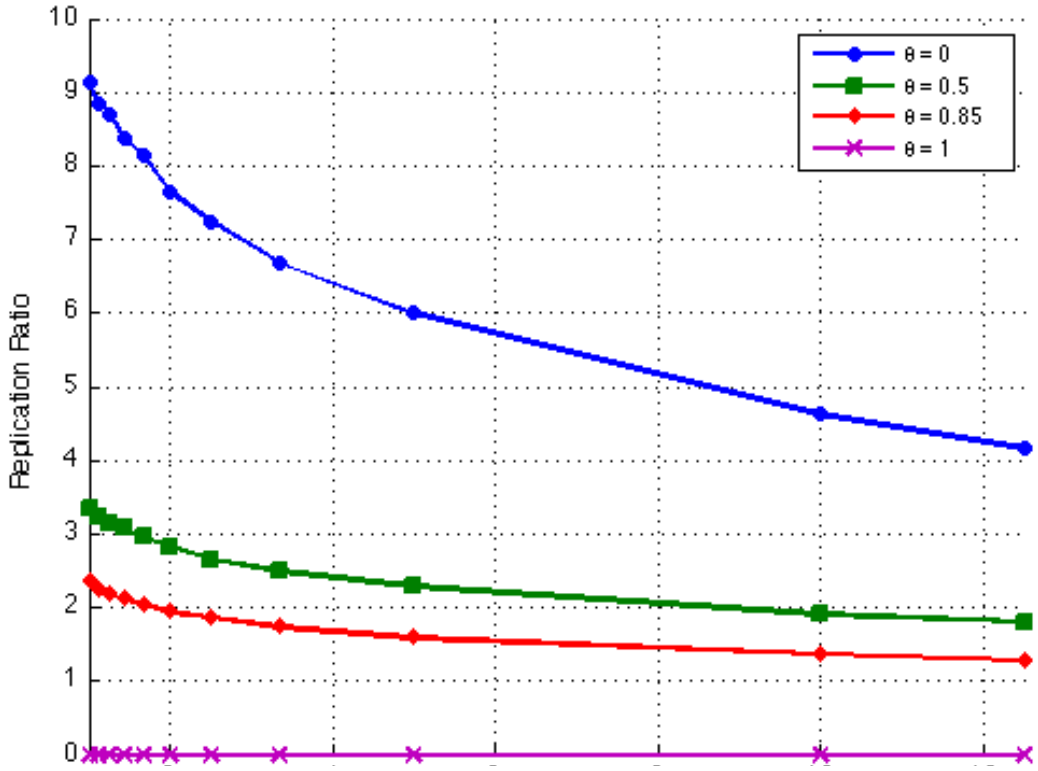
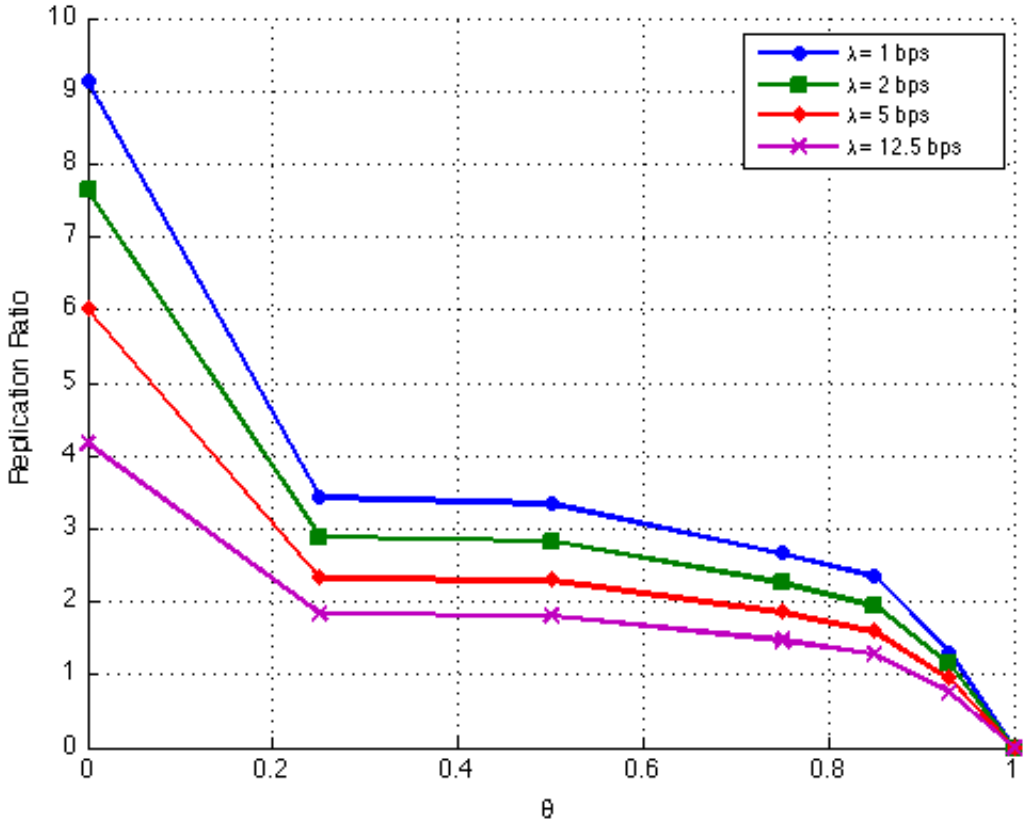


Grafico 4:



I grafici 3 e 4 confermano quanto detto sinora in termini di repliche. L'approccio binary spray limita a 10 il valore massimo del Replication Ratio, il quale decresce all'aumentare di Θ: quando Θ=1 non vengono generate repliche perché i pacchetti vanno consegnati direttamente alla destinazione.

Al crescere di λ il Replication Ratio cala, perché la maggior quantità di traffico rende più difficile lo scambio di informazioni tra i nodi, e di conseguenza diminuisce il numero di trasmissioni che vanno a buon fine.

Grafico 5:

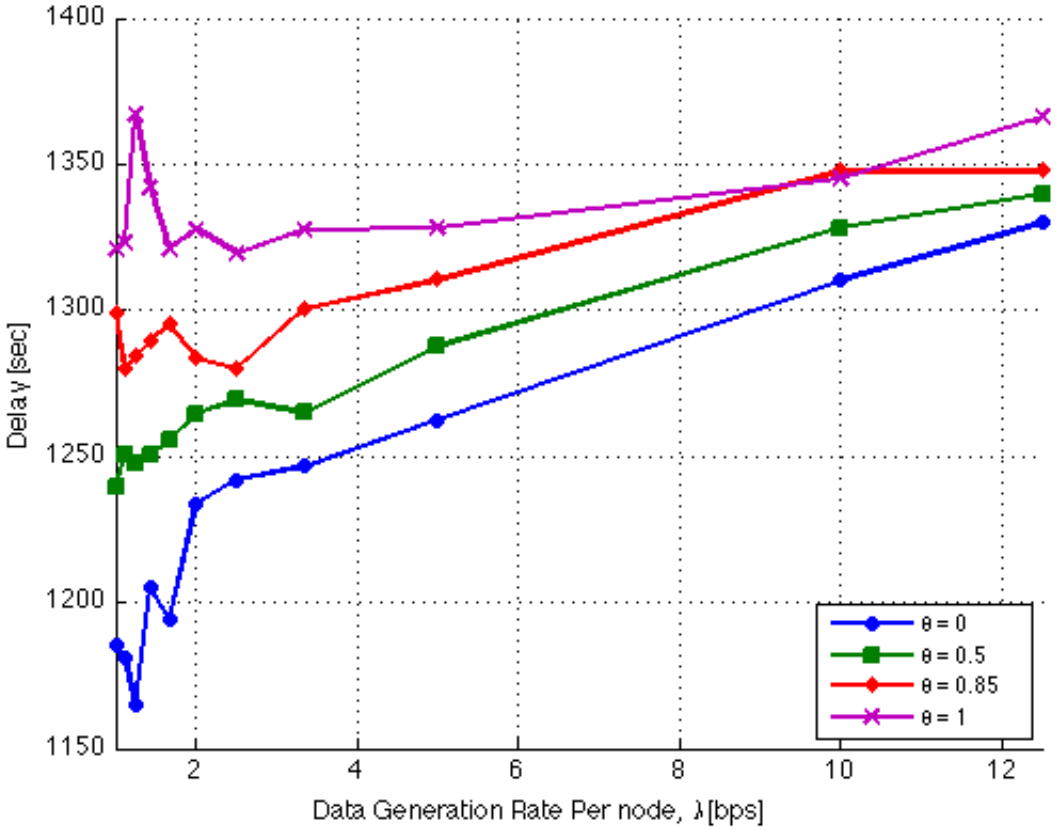
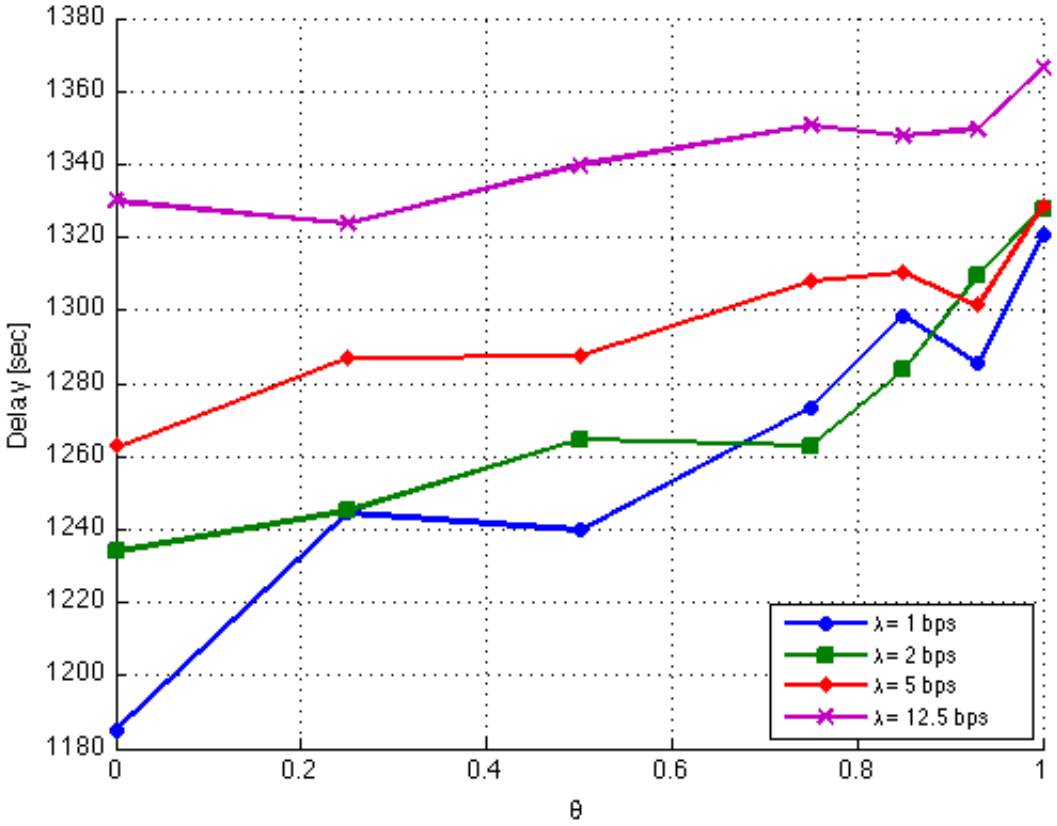


Grafico 6:



I grafici 5 e 6 sui ritardi di consegna, mostrano come questi aumentino sia al crescere di Θ che di λ. A valori bassi di entrambi i parametri, corrisponde una rete in grado di gestire bene il traffico, pur generando più repliche, ne deriva quindi una buona capacità di consegna dei pacchetti in tempi rapidi. Al crescere di Θ e λ i valori dei ritardi aumentano convergendo, questo perché le code si riempiono e i nodi trovano più difficoltà ad inviare pacchetti, portando ad un'inevitabile crescita del Delivery delay.

Il comportamento un po' anomalo del caso in cui $\lambda = 1$ bps, è dovuto al fatto che in condizioni di scarso traffico, il numero di pacchetti che concorrono al calcolo della media del ritardo di consegna è ridotto, conducendo a valori con intervalli di confidenza più ampi. Nel complesso, le prestazioni in termini di ritardo per il caso $\lambda = 1$ bps sono analoghe a quelle degli altri casi: per valori di Θ sufficientemente bassi, il processo di flooding concorre alla rapida consegna dei pacchetti, con conseguente diminuzione del ritardo; per valori di Θ più prossimi a 1, il numero di

copie generate per ogni pacchetto è minore (vedi grafico 4), ma il ritardo è corrispondente è maggiore.

I risultati delle simulazioni hanno permesso di confrontare il protocollo ai casi limite in cui può essere usato. L'utilizzo della tecnica del binary spray per limitare il numero di copie di ciascun pacchetto nella rete rende lineare il comportamento dell'algoritmo: al variare dei parametri infatti cambiano i valori, ma gli andamenti delle curve in molti casi sono simili.
Il protocollo offre diversi livelli prestazionali al variare di Θ e λ: questo è un possibile punto di forza, in quanto rende il protocollo adattabile a diversi scenari.

CONCLUSIONI E SVILUPPI FUTURI

Lo scopo di questa tesi è di fornire l'idea, i principi base e i dettagli implementativi di P-Spray, un protocollo di routing per reti acustiche sottomarine tolleranti al ritardo, concludendo con la valutazione delle performance in uno scenario simulato.
P-Spray è limitato dalla conoscenza a priori della statistica degli incontri tra i nodi, ma vista la sua semplicità può essere utilizzato in qualsiasi scenario: non c'è alcun vincolo sulla quantità di nodi che possono partecipare alla rete, sulla loro collocazione, sulla presenza o meno di sink ecc.
Il protocollo segue l'approccio replication-based, che nelle reti DTN garantisce buone percentuali di consegne corrette a scapito dell'impiego di risorse; P-Spray interviene proprio in tal senso, in quanto controlla le repliche dei pacchetti tarandole rispetto alla probabilità che il destinatario della replica incontri la destinazione dei pacchetti, e che quindi la replicazione sia utile. A contribuire ulteriormente al contenimento del traffico nella rete, vi è il meccanismo di replica di tipo binary spray, con un numero massimo di copie consentite pari al numero dei nodi presente nella rete.
In generale, considerando valori di interesse pratico, ad esempio con il Data Generation Rate di 2 bps e una soglia di probabilità di 0.75, il protocollo ha mostrato buone prestazioni, che comunque sono adattabili in base al tipo di esigenze della rete.
Come estensione futura del presente lavoro si potrebbe:

- ✓ permettere ai nodi di costruire in tempo reale la tabella delle statistiche degli incontri nell'arco della simulazione;

- ✓ progettare un opportuno meccanismo di controllo di errore.

Bibliografia

[1] Ethem M. Sozer, Milica Stojanovic, and John G. Proakis, "Underwater Acoustic Networks", IEEE *Journal of Oceanic Engineering*, vol. 25, no. 1, January 2000.

[2] Milica Stojanovic, “On the Relationship Between Capacity and Distance in an Underwater Acoustic Communication Channel”, OCEANS 2008 - MTS/IEEE Kobe Techno-Ocean, May 2008.

[3] R. Urick, Principles of Underwater Sound. McGraw-Hill, 1983.

[4] E. Sozer, M. Stojanovic, and J. Proakis, "Initialization and routing optimization for ad hoc underwater acoustic networks," in Proc. OP-NETWORK, Washington, DC, Sep. 2000.

[5] D. Pompili, T. Melodia, and I. Akyildiz, "Routing algorithms for delay insensitive and delay-sensitive applications in underwater sensor networks," in Proc. Annual International Conference on Mobile Computing and Networking (MobiCom), Los Angeles, CA, USA, Sep 2006.

[6] Michele Zorzi, Paolo Casari, Nicola Baldo, and Albert F. Harris III, “Energy-Efficient Routing Schemes for Underwater Acoustic Networks”, IEEE Journal, vol. 26, no. 9, December 2008.

[7] C.E.Perkins and E.M.Royer, “Ad-hoc On-demand Distance Vector routing,” Proceedings of the Second lEEE Woihhop on Mobile Computing Systems and Applications, 1999, WMCSA ‘99., pp. 9&100, 25-26 Feb. 1999.

[8] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks” Mobicom99, August 1999.

[9] M. Zorzi, R.R. Rao, “Geographic Random Forwarding (GeRaF) for ad hoc and sensor networks: multihop performance”, IEEE Transactions on Mobile Computing, vol.2, no.4, pp. 337-348, Oct.-Dec. 2003.

[10] P. Xie, J.-H. Cui, and L. Lao. Vbf: Vector-based forwarding protocol for underwater sensor networks. Proceedings of IFIP Networking, May 2006.

[11] Hai Yan, Zhijie Jerry Shi, and Jun-Hong Cui, “DBR: Depth-Based Routing for Underwater Sensor Networks”, Department of Computer Science anpossibile d Engineering University ofConnecticut.

[12] Edward A. Carlson, Pierre-Philippe Beaujean and Edgar An, "Location-Aware Routing Protocol for Underwater Acoustic Networks", OCEANS 2006, page(s): 1 - 6, 2006.

[13] Milica Stojanovic, ”On the Relationship Between Capacity and Distance in an Underwater Acoustic Communication Channel”, Massachusetts Institute of Technology.

[14] N. Baldo, F. Maguolo, and M. Miozzo, “A new approach to simulating PHY, MAC and Routing,” in Proc. ACM International Workshop on NS-2, Athens, Greece, Oct 2008.

[15] R.Coates, Underwater Acoustic Systems, New York: Wiley, 1989.

[16] R. Otnes, A. Asterjadhi, P. Casari, M. Goetz, T. Husøy, I. Nissen, K. Rimstad, P. van Walree, M. Zorzi, *Underwater Acoustic Networking Techniques*, SpringerBriefs in Electrical and Computer Engineering, DOI: 10.1007/978-3-642-25224-2_1, 2012.

RINGRAZIAMENTI

Un sentito ringraziamento va a Paolo Casari e a Saiful Azad per il costante supporto, l'aiuto e la grande disponibilità nell'arco della realizzazione del progetto di tesi. Grazie anche a tutti gli amici - colleghi del Signet, per avermi ben accolto e fornito sempre preziosi consigli.

Ringrazio infine la mia famiglia e i miei amici, ovvero coloro che sempre mi hanno sostenuto, dato fiducia, e regalato grandi momenti.